IB04/03266

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

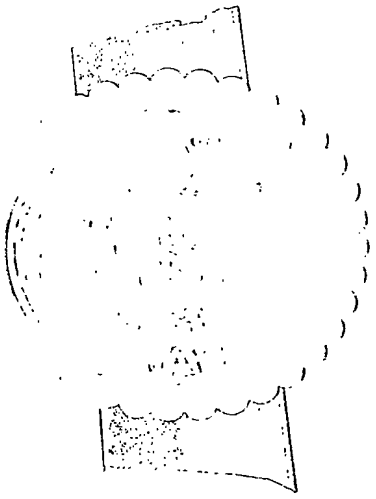
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年１０月　６日

出願番号
Application Number: 特願２００３－３４６４４２

[ST. 10/C]: [ＪＰ２００３－３４６４４２]

出願人
Applicant(s): カテナ株式会社



２００４年１０月２５日

特許庁長官
Commissioner,
Japan Patent Office

小川洋

出証番号　出証特２００４－３０９６１２６

【書類名】　特許願
【整理番号】　DCT03045
【提出日】　平成15年10月 6日
【あて先】　特許庁長官殿
【国際特許分類】　F01B 23/12
　　F01C 17/00
【発明者】
　【住所又は居所】　東京都多摩市落合１丁目１５番２号　カテナ株式会社内
　【氏名】　荒井　攻
【特許出願人】
　【識別番号】　396023362
　【氏名又は名称】　カテナ株式会社
　【代表者】　小宮　善継
【代理人】
　【識別番号】　100110559
　【弁理士】
　【氏名又は名称】　友野　英三
【手数料の表示】
　【予納台帳番号】　164782
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1

【書類名】特許請求の範囲
【請求項１】
ソフトウェア開発要望に係る基本構造に属する単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係順序関係から第１の隣接行列を生成し、
　前記基本構造同士の連結関係を規定する経路作用要素を単語とみなし、かかる単語の定義式及び実行条件を第２の隣接行列として表し、
　前記第１及び第２の隣接行列からシステム全体を律する構造を成立させ、
　前記成立した構造に対して探索を施して前記単語単位プログラム群を並び替える
　ことを特徴とするソフトウェア開発前処理方法。
【請求項２】
前記探索はトポロジカル・ソートによることを特徴とする請求項２記載のソフトウェア開発前処理方法。
【請求項３】
ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係を有向グラフで表すことを特徴とするソフトウェア制御方法。
【請求項４】
ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係に係る集合が閉じる単位で基本構造単位を構成し、
　前記構成された基本構造単位をノードとして捉えた有向グラフで表すことを特徴とするソフトウェア制御方法。
【請求項５】
ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係から隣接行列を生成し、
　前記生成された隣接行列に対して探索を施し該探索の結果をもとに前記単語単位プログラム群を並び替え、
　前記並び替えられた前記単語単位プログラム群をＬｙｅｅ方法論に適用してコードを生成する
　ことを特徴とするソフトウェア開発方法。
【請求項６】
ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係から隣接行列を生成する隣接行列計算部と、
　前記生成された隣接行列に対して探索を施し該探索の結果をもとに前記単語単位プログラム群を並び替える探索部と、
　前記並び替えられた前記単語単位プログラム群をＬｙｅｅ方法論に適用してコードを生成する生成部と
　を具備することを特徴とするソフトウェア開発装置。

【書類名】明細書
【発明の名称】ソフトウェア開発前処理方法、ソフトウェア制御方法、ソフトウェア開発方法並びにソフトウェア開発装置
【技術分野】
【０００１】
本発明は、ソフトウェア開発に係り、特に、数学的考え方にたったソフトウェア開発前処理方法、ソフトウェア制御方法、ソフトウェア開発方法並びにソフトウェア開発装置に関する。
【背景技術】
【０００２】
一般にプログラムは、次のような３つの部分によって構成される。
（１）領域の宣言、メモリの割り当てとそれに関連した定数及び変数のアドレス
（２）制御の順序を表す文
（３）メモリ割り当て領域に与えられる値を決定するための実行条件文と計算実行文

ソフトウェアが開発されるとき、ユーザは、主に（３）にのみ関係する。他の部分については、システムエンジニア及びプログラマが責任をもつ。人間は、コンピュータそれ自身を制御しているプログラムによって、画面及びキーボードからデータを入力し、ファイル中のデータを読み、またファイルにデータを書きこむ。
この部分に使用する装置を動かすプログラムも要件に依存せず、システムエンジニア及びプログラマが責任をもつ。このうち、特に、（２）の部分には要件に依存しない固定の(一定の)構造が存在する。したがって、要件に依存する（３）を決定する（すなわち、定義する）だけで、自動的にプログラムを生成することができる。
この構造体は Ｌｙｅｅ（登録商標） の方法論の上では「プログラムの構造」といわれる（特許文献１乃至６、非特許文献３及び４参照）。

我々は通常、ホーア論理（Hoar Logic）やλ計算のような論理的な関係を用いてプログラム構造を表現する（非特許文献１及び２参照）。 このスタイルのプログラム構造の表現は、それらのプログラムの行動的構造を理解することの助けになる。これにより、特に宣言型(スタイル)プログラムの場合に、プログラムを理解し、分析することが可能になる。しかしながらこのようなアプローチを通してされた分析は、論理表現の持つ複雑さゆえに、意味論を検証したり、そして／あるいは、意味論を修正したりすることの困難さを顕在化する。

【特許文献１】国際公開第９７／１６７８４号パンフレット
【特許文献２】国際公開第９８／１９２３２号パンフレット
【特許文献３】国際公開第９９／４９３８７号パンフレット
【特許文献４】国際公開第００／７９３８５号パンフレット
【特許文献５】国際公開第０２／４２９０４号パンフレット
【特許文献６】特開２００２－２０２８８３号公報
【非特許文献１】グリン・ウィンスケル著、「プログラミング言語の公式意味論」、ＭＩＴプレス、１９９３年
【非特許文献２】ハンネ・リース・ニールソン、フレミング・ニールソン著、「アプリケーションについての 意味論　公式紹介」、ジョン・ワイリー・アンド・サンズ、１９９２年
【非特許文献３】根来　文生著、「Lyee ソフトウェアの原理」、２１世紀（ IS2000 ）における情報社会についての国際会議会報、日本、会津、２０００年１１月５日-８日、pp441 - 446
【非特許文献４】根来　文生著、「ソースコード生成についての高密度処理方法」、システミクス、サイバネティクス及びインフォマティクスについての第５回世界多極会議会報、（ SCI2001 ）、米国、オーランド、２００１年７月２２日-２５日

【発明の開示】
【発明が解決しようとする課題】
【０００３】
このように、従来の論理的な関係を用いたプログラム構造についての表現は、特に宣言型(スタイル)プログラムの場合に、プログラムを理解し、分析することが可能になる。しかしながらこのようなアプローチを通してされた分析は、論理表現の持つ複雑さゆえに、意味論を検証したり、そして／あるいは、意味論を修正したりすることの困難さを顕在化するものであった。
【０００４】
本発明は、前述のような従来の技術が有していた諸問題点を解決しようとするものであって、当該プログラムが複雑な論理表現を持っていても、意味論を容易に検証し、修正し得る方法論を提供することを目的とするものである。
【０００５】
本発明は、これにより、Ｌｙｅｅ（登録商標）方法論によるソフトウェア開発において、より高効率、高速、高パフォーマンスを実現することを目的とする。
【０００６】
さらに本発明は、上記のような目的を達成できるソフトウェア開発前処理方法、ソフトウェア制御方法、ソフトウェア開発方法並びにソフトウェア開発装置を提供することを目的とする。
【０００７】
また、本発明は上記と併せて、Ｌｙｅｅ（登録商標）方法論に係るプログラム構造を数学的に表現し、その正当性を示すこと、及び、その順序性のない性質を示すことを目的とする。
【課題を解決するための手段】
【０００８】
上記課題を解決するために、本発明は、ソフトウェア開発要望に係る基本構造に属する単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係順序関係から第１の隣接行列を生成し、該基本構造同士の連結関係を規定する経路作用要素を単語とみなし、かかる単語の定義式及び実行条件を第２の隣接行列として表し、該第１及び第２の隣接行列からシステム全体を律する構造を成立させ、該成立した構造に対して探索を施して前記単語単位プログラム群を並び替えることを特徴とする。
【０００９】
さらに本発明は、ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係を有向グラフで表し、該表された有向グラフに対して探索を施し、該探索の結果をもとに当該単語単位プログラム群を並び替えることを特徴とする。
【００１０】
また本発明は、ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係から隣接行列を生成し、該生成された隣接行列に対して探索を施し、該探索の結果をもとに当該単語単位プログラム群を並び替えることを特徴とする。
【００１１】
ここで、「探索」とは、（後述する）トポロジカル・ソート、（Ｔｏｐｏｌｏｇｉｃａｌ　Ｓｏｒｔ）、（後述する）深さ優先探索（Ｄｅｐｔｈ－Ｆｉｒｓｔ　Ｓｅａｒｃｈ）、幅優先探索（Ｂｒｅａｄｔｈ－Ｆｉｒｓｔ　Ｓｅａｒｃｈ）、反復深化法（Ｉｔｅｒａｔｉｖｅ　Ｄｅｅｐｉｎｇ）、発見的探索（Ｈｅｕｒｉｓｔｉｃ　Ｓｅａｒｃｈ）、ヒル・クライミング（Ｈｉｌｌ　Ｃｌｉｍｂｉｎｇ）、最適優先探索（Ｂｅｓｔ－Ｆｉｒｓｔ　Ｓｅａｒｃｈ）、オイラー（Ｅｕｌｅｒ）の一筆書き、ダイクストラ（Ｄｉｊｋｓｔｒａ）法等を含む各種アルゴリズムをいう。
【００１２】

また本発明は、ソフトウェア開発のユーザ要件中の単語についての有意性規定関係から隣接行列及び状態ベクトルを生成し、当該隣接行列と状態ベクトルとを所定の公理に基づき計算し、該計算の結果から、単語の状態を把握することを特徴とする。

【００１３】

さらに本発明は、ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係を有向グラフで表すことを特徴とする。

【００１４】

ここで、順序関係としては半順序及び全順序が考えられる。

【００１５】

また、ここで用いる「ソフトウェア」とは広義の意である。即ち、ユーザからの開発要望及び当該要望にあった目的プログラムの双方を包含する概念である。

【００１６】

また本発明は、ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係に係る集合が閉じる単位で基本構造単位を構成し、該構成された基本構造単位をノードとして捉えた有向グラフで表すことを特徴とする。

【００１７】

さらに本発明は、ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係から隣接行列を生成し、該生成された隣接行列に対して探索を施し該探索の結果をもとに当該単語単位プログラム群を並び替え、該並び替えられた当該単語単位プログラム群をＬｙｅｅ方法論に適用してコードを生成することで求めるソフトウェアを得ることを特徴とする。

【００１８】

また本発明は、ソフトウェア開発要望に係る単語ごとのエリアに値を求めるための単語単位プログラム相互の順序関係から隣接行列を生成する隣接行列計算部と、該生成された隣接行列に対して探索を施し該探索の結果をもとに当該単語単位プログラム群を並び替える探索部と、該並び替えられた当該単語単位プログラム群をＬｙｅｅ方法論に適用してコードを生成する生成部とを具備することを特徴とする。これら各部は総て自動化できるので、結局繰返し処理の排除→ソフトウェア生成まで一貫して自動処理が可能となり、ソフトウェア生産速度、効率は各段に向上する。

【００１９】

ここで、（後述する）「有向グラフ作成部」とは、ソフトウェア開発要望者であるユーザの発する要件に含まれるそれぞれの単語をノードによって表し、入力単語と状態単語とを区別しつつa ≦ b（aからｂが作られることを意味する記号として「≦」を定義する。以下同じ。）という始点と終点の間の半順序関係（または全順序関係）を矢印によって表した有向グラフを作成する機能を実現する単位をいう。例えば、こうした機能をプログラム化したソフトウェアとしてもよいし、或いは当該ソフトウェアをコード化したものをＲＯＭ（ＲｅａｄＯｎｌｙＭｅｍｏｒｙ）に書きこんだものであってもよい。

【００２０】

ここで、「入力単語」とは（後述の）数式３におけるf_i を持っていない最終の終点単語をいい、「状態単語」とは入力単語でない単語をいう。なお、以下では、数式を指標するにあたり、数式の見出し中では「数」の表題に番号を付したものをもってするが、文章中指標するにあたっては、「数式１」のようにいうこともある。

【００２１】

また、「隣接行列計算部」とは、単語単位のプログラム相互の関係を（後述の）数式３形式で表現し、それを行列の形式で再表現した形態である隣接行列を計算して得る機能を実現する単位をいう。例えば、こうした機能をプログラム化したソフトウェアとしてもよいし、或いは当該ソフトウェアをコード化したものをＲＯＭ（ＲｅａｄＯｎｌｙＭｅｍｏｒｙ）に書きこんだものであってもよい。

【００２２】

さらに、「探索部」とは、前述の各種手法を包含するアルゴリズムである探索を一般に

実現する機能を有する。例えば、こうした機能をプログラム化したソフトウェアとしてもよいし、或いは当該ソフトウェアをコード化したものをＲＯＭ（ＲｅａｄＯｎｌｙＭｅｍｏｒｙ）に書きこんだものであってもよい。

【００２３】

また、（後述の）「トポロジカル・ソート部」とは、上記で得られた有向グラフに対して（後述の）深さ優先探索を行い、行きついたところから探索経路を戻るときにノードを取得してゆくことでノード列（数列）を整列する機能を実現する単位をいう。例えば、こうした機能をプログラム化したソフトウェアとしてもよいし、或いは当該ソフトウェアをコード化したものをＲＯＭ（ＲｅａｄＯｎｌｙＭｅｍｏｒｙ）に書きこんだものであってもよい。

【００２４】

ここで、「深さ優先探索」とは、
始点であるノードを出発してノード間の連結がとれているノードを訪問が重複しないように進み、
行く場所がなくなったら行く場所のあるノードまで戻り（再帰）、
戻ったノードから再び（１）の要領で進み、
進むところがなくなったら終了する、
というアルゴリズムをいう。

【００２５】

上記で規定される本願に係る発明では、繰返しを回避するので、より高効率、高速、高パフォーマンスを備えたソフトウェア開発を実現することが可能となる。

【００２６】

さらに上記で規定される本発明では、繰返しをなくす前処理を自動で行い、こうして得られた前処理済の単語単位のプログラムをもとにＬｙｅｅ（登録商標）方法論により自動的に目的プログラム生成を行う。つまり、ユーザ要件の洗練化（整列）から目的プログラムの生成に至るまで自動化することが可能となる。これにより、ソフトウェア生産の大幅な効率向上、生産性向上、品質向上等、ソフトウェア産業上に大きな効果をもたらす。

【発明の効果】

【００２７】

本発明によれば、Ｌｙｅｅ（登録商標）方法論によるソフトウェア開発において、より高効率、高速、高パフォーマンスを実現することができる。

【発明を実施するための最良の形態】

【００２８】

以下、図面を参照しながら、本願の発明の具体的な実施形態について説明する。

【００２９】

第I章で、本発明の前提となる、技術および概念について述べる。第Ｉ章のＩ－１では、本発明が改良を施す対象であるＬｙｅｅ（登録商標）方法論によるソフトウェア構造（単語単位のプログラム構造）と、Ｌｙｅｅ（登録商標）の特徴的な概念である「状態」の概念について概略を述べる。Ｉ－２では、単語単位のプログラムを数学的構造モデルによって説明する。Ｉ－３で、単語単位のプログラムによる値の生成を一巡処理で完了するための技術トポロジカルソートについて述べる。トポロジカルソート技術については、ＰＣＴ／ＪＰ０３／０９５９１で特許出願済みである。

【００３０】

第ＩＩ章で、本発明の具体的な実施形態である、トポロジカルソートをＬｙｅｅ方法論のシステム全体に施すためのシステム構造の単一化について述べる。第ＩＩ章のＩＩ－１では、Ｌｙｅｅのシステム構造について述べる。ＩＩ－２で、Ｌｙｅｅのシステム構造の単一化技術について述べる。

Ｉ章

Ｉ－１．単語単位のプログラムとは何か

Ｉ－１－（１）単語単位のプログラムとは

Ｌｙｅｅ（登録商標）方法論によるソフトウェアは、単語単位のプログラムを基本にして構成されている。単語単位のプログラミングでは、要件は単語の集合によって表され、単語は宣言文を使って定義される。単語単位のプログラムは、２種類の主なコンポーネントで構成されている。１つのコンポーネントは単語のコンポ-ネントである。単語単位のプログラミングでは、要件は、単語の名前、単語の計算式の定義、単語の計算式の条件、単語の属性（入出力、データタイプ、その他の属性）を含んだ単語の集合とし、代入文ではなく宣言文の形式で定義される。

たとえば、Ｓｗが単語ｗを定義する宣言文であるとすると、その要件の例は表１のようになる。

【００３１】

【表１】

	単語名	属性	条件	定義
単語	*a*	出力	*d*	*b* ＋ *c*
単語	*c*	入力 整数		
単語	*b*	出力	*c* ＞ 0	2 × *c* ＋ 5
単語	*d*	入力 ブール		

もう１つのコンポーネントは制御プログラムのコンポーネントである。制御プログラムは、それ以上実行を繰返しても、いかなる単語の値も変化しない状態になるまで、単純に単語単位プログラムの実行の繰り返しを行う。単語ａの定義は単語ｂを使用しており、単語ｂは単語ａの後に与えられているが、ユーザはこれらの定義が実行される順序（制御ロジック）を指定する必要がない。一方、手続き型プログラミングの場合は当然ながら、宣言型プログラムミングの一種であるロジカルプログラミングの場合でも、プログラムの一行の実行順序は実行結果に影響を与える。

【００３２】

表１の例を使って、この処理の動きを説明する。

【００３３】

最初に、単語ａが実行される。単語ｄ、ｂ、ｃはまだ実行されておらず、従って単語ａは値を決定することができない。次に、入力単語ｃが実行される。単語ｃは実行条件がないので、値が決定される。例えば、単語ｃの値が＋３だとすると、単語ｂは、実行条件ｃ＞0が真となるので、定義　２×c＋５　が実行可能になり、単語ｂの値は＋１１と計算され、値の出力が可能となる。単語ｄは入力単語であり、実行条件がないので、ｄ＝真である。これですべての単語が処理されたが、単語ａの値は未決定のままである。結果として、プログラムは最初に戻って再び処理を実行する。

【００３４】
２回目の実行時、ｄ　＝真、単語ｂと単語ｃはそれぞれ値が＋１１と＋３に決定しているので、単語ａは値＋１４を得て、その値が適切な装置に出力される。単語ｃ、ｂとｄの値は前に計算されているので、これらの単語の値のすべては決定済みである。従って、２回目の実行で、４つの単語の総ての値が決定したことになる。
【００３５】
次に、「状態」概念について説明する。単語のデータタイプは、コンピュータ上で扱い可能な、数、ブール（boolean）、文字、その他である。論理的順序を制御するためには、それぞれの単語に存在する「状態」の概念の導入が必要となる。実際、単語の状態とは、複数の状態：すなわち、真、偽、そして未決定のいずれか１つとなる。それぞれの意味は次の通りである。
真：値が決定され、かつユーザ要件を満たす。
偽：値が決定されるが、それはユーザ要件を満たさない。
未決定（又は空）：値が決定されていない。

単語の最初の状態は常に未決定で、それは単語の値がまだ転送（move）されていないか、あるいは計算されていなかったことを意味している。名前によって定義された単語を終点単語、そして、別の単語を定義するために、計算条件や計算式に使われている単語群を始点単語と呼ぶ。要件中の総ての単語が１回実行されることを一巡処理と呼ぶ。
【００３６】
単語の値が決定され、その状態が真となるのは、その単語の始点単語の状態がすべて真となったときである。始点単語として使われていない単語の状態は、終点単語の状態に関与しない。一巡処理が繰り返されることを、繰返し処理（又は、繰返しプロセス）と呼ぶ。単語の状態に変化がなくなるまで繰り返し処理を続けることにより、単語を出力に用いることが可能となる。各単語の状態は、真、偽になるか、あるいは未決定のまま残るか、のいずれかである。もし未決定の単語があるなら、そのシステムに入力単語が存在するかどうか、閉じたループがあるかどうかをチェックする必要がある。
【００３７】
閉じたループは、単語が直接的または間接的に自身で自身を定義しているか、要件が完全でないことを意味する。実行条件は排他的でなければならず、また否定は明示的に定義されていなくてはならない。すなわち、条件の総和が重なりのない全体を構成（言い換えれば、条件の総和が１となり、１以上でも１以下でもない）しなければならない。
【００３８】
このように、単語単位のプログラミングは、ソフトウェアの制御ロジック部分を自動的に生成する。

Ｉ－１－（２）個々の単語のプログラム構造

図１は、前述した単語単位のプログラムの構造を示したフローチャートである。同図をもとにプログラムの構造及び動作について説明する。図1のαは、単語名を示す。
（１）ボックス１
まず、statement（宣言文）を遂行するべきかどうか判断する。もし単語αの状態が空でないなら、以下を実行せず終了し（３０１）、空ならボックス２へ進む。
（２）ボックス２
もし始点単語の状態が総て真であるときは、単語の定義式（計算式）を遂行して、その結果を一時的な領域に記憶する。従って、一時的な領域が必要となる。始点のいずれかが真以外の場合は、定義式を遂行しない。
（３）ボックス３
定義式を実行した、または不実行の結果、単語αに有意な（ユーザ要件に適合する）値が決定されたかどうかを判断する。総ての始点単語が真であるとき、単語の値は決定され

る。しかし、たとえ総ての始点単語が真であるとしても、ユーザの要求条件によっては単語の値は偽とされるかもしれない。単語に有意な値が決定されるときはボックス４へ、単語に値が決定されない、又は決定された値が偽とされるときはボックス５へ移行する。

（４）ボックス４

決定された単語αの値がユーザの要求条件に適合するときには、通常領域に結果を記録する。従って、通常領域が必要となる。この場合には、単語αの状態は真である（３０３）。

（５）ボックス５

ボックス３で単語の値が未決定、又は決定された値が偽とされた場合はここに至る。ボックス５では、単語がこのうちどちらであるかを判断する。

単語の値がユーザ要求条件によって偽となるか、あるいは始点の値が偽の場合は、単語の状態は偽である。単語の状態が偽となる場合はボックス６へ、単語の値が偽でない場合はボックス７へ移行する。

（６）ボックス６

値が偽であることを示すフラグをセットする。このようなフラグは偽のプロセスとしてすべての単語に必要である。または、状態が偽であることを意味する値を通常領域に記憶する。このとき単語の状態は偽となる（３０５）。

（７）ボックス７

繰り返し処理を実行するかどうかを判断するために再帰（繰返し）カウンタに１を加える。従って、再帰カウンタが必要となる（非特許文献３参照）。このとき単語の状態は空となる（３０７）。

Ｉ－１－（３）全体のシステムを制御するプログラムとその構造

システム全体を制御するプログラムがパレット連鎖関数と呼ばれる（非特許文献３参照）。適切に繰り返し処理をする制御プログラムはこのプログラムの中に入れられる。図１で説明したように、もし１つの始点単語の状態が偽であるなら、終点単語の状態が確実に偽になる。たとえ始点単語のすべての状態が真であるとしても、ユーザ要求条件について偽があるときには、終点単語の状態が偽になる。したがって、繰り返しがなされるかどうかを判断するためには、ただ空（未決定）の始点単語が定義式に含まれるか否かを判断すればよい。

【００３９】

もし空の始点単語が１つでもあれば、繰り返し処理は実行される。しかしながら、有向グラフによってそれを表現するならば、閉じられたループが含まれるときには、終点単語の状態が空から変化しない。また、このシステムが始点単語を持っていないとき、終点単語は空から変化しない。そのために、このような場合には、繰り返しプロセスは終了せず永久のループに入る。このような状況を回避するために、プログラムのすべての単語Ｌ(α)を制御するパレット連鎖関数のプログラム（Φ）にある再帰カウンタを制御するために用いられるプログラムコードがある。

【００４０】

単語が空になるたびに、それぞれの単語のプログラムが再帰カウンタに１を加える。一連の繰り返しプロセスを終了する条件として、この再帰カウンタの値が前回の繰り返し処理での値と同じであるということを用いる。この状態を「状態変化はない」という。

【００４１】

ロジカルプログラミングでは、特に非手続き型プログラミング言語の一種である標準的なプロログ（Prolog：非手続き型プログラミング言語）では、バックトラック（再実行）が永久ループに陥る。他の改良型プロログは、待機（wait）やフリーズを使って永久ループを回避する。単語単位のプログラムでは、前述のメカニズムを使ってこのようなループを回避する。

Ｌｙｅｅ（登録商標）方法論で用いるパレット連鎖関数プログラム（Φ）の例を以下に

示す。「Crtの値は状態が未決定（空）である単語L(α)の数である」をループの変数プロパティとするというプロパティを用いることによって、このプログラムはホア（Hoare）の理論で証明することができる。下記の例はｘ、ｙ及びｚの３つの単語がある場合である。

［Φのプログラムコード］

```
PCtr = 1      /* PCtr = 前回の繰り返し処理の再起カウンタ */
Ctr =
0       /* Ctr = 再起カウンタ */
While
PCtr &supl; Ctr do
      PCtr = Ctr
      Ctr = 0
L(x)       /* 各単語プログラムは状態と繰り返しの制御を行う以下のような関数（機能）を持っている;
もし始点単語が空なら、単語L(α)の状態は未決定で Ctr = Ctr+1
そうでないなら、単語L(α)の状態は決定済*/
L(y)
L(z)
End
/* Ctrの値は状態が未決定である単語L(α)の数*/
```

このプログラムは以下の意味になる：
プログラムが終了する。
終了したとき、変数Ctrの値は、状態が未決定である単語L(α)の数である。

プログラムが停止したとき、単語の定義式に従って、このシステム中に始点単語を持つ単語Ｌ(α)の状態は、通常の場合真か偽が割り当てられる。単語αの状態は、次のような場合は、未決定（空）である：
１）単語αが閉じたループの中か、閉じたループの後にある場合。
２）単語αが排他的条件を持っていない場合。
排他的とは、もし
【００４２】
【数１】

$$((x \geq y)\ \mathrm{or}\ (x < y))$$

が常に真であり、かつ、
【００４３】
【数２】

$$((x \geq y\ \mathrm{and}\ (x < y))$$

が常に偽であれば、ｘ
とｙ
は排他的である、として定義される。
３）単語αの始点単語が同じシステムにない場合。

Ｉ－２．単語単位のプログラムの数学的考察

単語単位のプログラムのプログラム構造はホアの定理によって証明できる。しかしながら、そのようなホアの定理のアプローチを通しての分析では、その論理表現のスタイルの複雑性のために、単語単位プログラムの意味論を検証し、改良を加えることが困難である

。そこで、有向グラフ技術に起源したもう１つのアプローチである隣接行列を用いて、単語単位プログラムによるプログラム構造を数学的構造モデル化し、本発明を説明する。この章では、まず単語単位のプログラムによるプログラム構造を、数学的構造モデルによって表し、その特徴を述べる。すなわち、有向グラフモデルと隣接行列モデルである。

Ｉ－２－（１）単語単位プログラムの関数表現

Lyee 要件の仕様をより形式的に扱うと、関数ｘを、１、２、…nの点における定数とその他の点のｘの値によって定義した方程式のシステムに表すことができる。

【００４４】

【数３】

$$x(i) = fi\ (\ x(1), x(2), \cdots, x(n)) \qquad i = 1,\ 2, \cdots,\ n$$

数式３は、単語がｎ個あるシステムにおける、ｉ番目の単語ｘの定義式を表す関数である。ｘ(i)は終点単語であるｉ番目の単語ｘを表す。fi (x(1),x(2),…,x(n))は、終点単語ｘ(i)　の定義式を表す関数で、定義式がシステム内の１からｎ番目の単語のいずれかによって構成されていることを示す。

数式３の　Ｌｙｅｅ　（登録商標）仕様は次のように書き直すことができる：

【００４５】

【数４】

$$x(i) = F\ (\ x(i)) \qquad i = 1, 2, \cdots, n$$

さらに簡易な形式では、

$x = F\ (\ x)$

である。

Ｉ－２－（２）本発明の原理：有向グラフ（directed graph）とその隣接行列

始点単語と終点単語の関係は、有向グラフによって表すことができる。有向グラフでは、各単語はノードによって、単語間の関係は矢印によって表される。矢印の出発点となるノードが始点単語、終着点となるノードが終点単語を表す。さらに、それはa ≦ｂ(単語aから単語ｂが作られる)という関係をもつ順序を定義する。

【００４６】

図２は、a ≦ｂ(aからｂが作られる)という関係（f）をもつ順序を説明するための有向グラフの例を示した図である。

【００４７】

同図に示すように、単語はa ≦ｂという関係を持つ半順序づけられた集合(セット)を構成する。即ち、同図の有向グラフは単語a及び単語ｂという要素及びこれらの要素間を関係付ける半順序関係（矢印）を構成要素とする集合とみることができる。

【００４８】

有向グラフは隣接行列によって表現することができる。ｎ個の単語の関係を表す隣接行列は、n個×n個のマトリックスで表され、これを隣接行列Ｆと呼ぶ。この隣接行列において、行見出しと列見出しには、それぞれn個の単語名が置かれる。行の見出しとなるときn個の単語はそれぞれ終点単語であり、列の見出しとなるときn個の単語はそれぞれ始点単語である。

【００４９】

i番目の単語の式（つまり、数式３によれば、単語x(i)を生成するプログラムf_i(x(i))）が、始点単語としてx_jを使うとき、隣接行列のi行目と始点単語x_j列の交点を表す隣接行列Ｆの要素f_{ij}(i=1,…,n; j=1,…,n)は１である。そうでないなら、０である。終点単語x_iを生成する式において、始点単語x_jが使用されているか否かを１か０で表すのである。（※「f_{ij}(i=1,…,n; j=1,…,n)」は、単語x(i)を生成するプログラムf_i(x(i))を構成する単語のうちの単語x(j)を指し、このときはiは１からｎ、ｊは１からnのいずれかである、ことを示している。）

各単語が数式３の形式で、下記の数式５のように定義される例１の単語の関係を、有向グラフ、隣接行列の２つの形態で表して具体的に説明する。

【実施例】

【００５０】

（例１）

入力単語（f_i(x(i))を持っていない最終の終点単語）をx_1及びx_2と置き、状態単語(入力単語でない単語、すなわちプログラムによって値を生成する単語)をx_3、x_4、x_5、x_6及びx_7と置く。これらの関係を上記数式３の形式で表すと下記の数式５のようになる。

【００５１】

【数５】

$$x(1) = a_0$$

$$x(2) = b_0$$

$$x(3) = f_3(x(4),\ x(5))$$

$$x(4) = f_4(x(2),\ x(5))$$

$$x(5) = f_5(x(1))$$

$$x(6) = f_6(x(3),\ x(7))$$

$$x(7) = f_7(x(4),\ x(6))$$

図３は、上記の例１の状態を表した有向グラフの例である。

【００５２】

即ち、同図に示す例においては、x_1及びx_2はf_i(x(i))を持っていない入力単語である。よって、いずれの矢印の終着点にもなっていない。x_3はx_4及びx_5から生成される。よって、x_3はx_4及びx_5から出発する矢印の終着点になっている。同様に、x_4はx_2及びx_5から、x_5はx_1から、x_6はx_3及びx_7から、x_7はx_4及びx_6から生成されることを表すように矢印が出発点と終着点が表されている。つまり、数式５で規定される状態と図３の有向グラフで表される状態とは等価である。

【００５３】

このとき例１の隣接行列は数式６のように表現される。

【００５４】

【数６】

$$F = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

先に述べたように、隣接行列Ｆの行見出しは終点単語であり、列見出しは始点単語である。各行または列がどの終点単語または始点単語を示すかをわかりやすくし、説明を簡便にするために、隣接行列には見出しを付した。

【００５５】

数式６の隣接行列Ｆを各行について展開して説明する。第１行目の終点単語x_1の行は、いずれの始点単語列との交点も０になっており、終点単語x_1は始点を持たないこと（すなわち入力単語であること）が示されている。終点単語x_2についても同様である。第３行目の終点単語x_3の行は始点単語x_4とx_5の列との交点が１、その他の始点単語列との交点が０になっており、終点単語x_3はx_4とx_5を始点単語として持つことが示されている。以下、終点単語x_4から終点単語x_7の終点単語についても、同様に、どの単語を始点単語として持つか（あるいは持たないか）が示されている。

【００５６】

数式６の隣接行列の各行は、以下のような数式でも表すことができる。左辺が行見出しとなる終点単語、右辺は、交点の値が１となった列見出しの始点単語を表す。右辺の「０」は、いずれの交点の交点の値も「０」であることを表す。

$x_1 = 0$

$x_2 = 0$

$x_3 = x_4$

$+x_5$

$x_4 = x_2 + x_5$

$x_5 = x_1$

$x_6 = x_3 + x_7$

$x_7 = x_4 + x_6$

ここで、「＋」は、後述するように、通常の四則演算における「加える」の意味でなく、本明細書中の単語の関係を表す数式においては、左辺の状態単語（終点単語）を決める半順序集合の始点単語が複数あるときにこれらを付加して左辺の単語を規定することを示す演算子（以下、「付加演算子」と呼ぶ）として用いている。

【００５７】

以上より、したがって、数式５で定義される単語の関係は、図３の有向グラフで表される関係及び数式６の隣接行列による数学的表現で記述される関係と等価であるということがいえる。

Ｉ－２－（３）隣接行列Ｆと状態ベクトルＸで表す終点単語のための状態モデル

前セクションでは、n個の単語の「終点単語の他の単語との関係」を表すためのモデルとして、有向グラフと隣接行列を考察した。次に、n個の単語の「終点単語の値の状態」を表すためのモデルについて述べる。まず、ｎ個の単語の始点単語としての状態を縦列のベクトルＸに記録する（ベクトルとは１つの列あるいは行しか持たないマトリックスである）。数式６の隣接行列に表された単語の例で説明すると、始点単語が表示される順序、x_1からx_7の順序に各始点単語の状態を表したベクトルXは、たとえば下記の数式７のように表すことができる。各行がどの始点単語を示すかをわかりやすくし、説明を簡便にするために、ベクトルには見出しを付した。

【００５８】

【数７】

$$X=\begin{matrix}x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\end{matrix}\begin{bmatrix}+1\\+1\\-1\\\text{null}\\\text{null}\\\text{null}\\\text{null}\end{bmatrix}$$

このとき、ベクトルXは、始点単語x_1とx_2の状態は「＋１」、始点単語x_3の状態は「－１」、x_4からx_7の始点単語の状態は「null」であることを示している。

【００５９】

単語を生成し得る値の状態は次のように３種類に設定される。

＋１：値は真であるとして決定済みである。

－１：値)は偽であるとして決定済みである。

空(null)：値は未決定である。

入力単語の初期値は以下の通りである。

１：値は真であるとして決定済みである。

（適切に入力されている状態）
－１：値は偽であるとして決定済みである。

（適切に入力されていない状態）
従って、たとえ数式３が実行されるとしても、状態は変化しない。
【００６０】
状態単語の初期値は空である：
空：値はまだ決定済みでない。

（移動していない、あるいはまだ計算されていない状態）

状態単語について考察するに、単語の関係を表す隣接行列Fの上に単語の状態を表す状態ベクトルXを作用させるということは当該単語の順序で数式３を横列で実行することである。順番にｎ個の式を使う処理は順序処理と呼ばれる。その始点単語のすべての状態が＋１であるとき、状態単語の状態が＋１（真）になり、状態単語の値は決定済みとなる。
【００６１】
始点単語ではない単語の状態はその式の終点単語の状態とは関係していない。順序処理が繰り返されることは繰り返し処理と呼ばれる。
たとえそれが繰り返されるとしても、「真」であるとして決定が済んでいない単語は、偽の状態（－１）になるか、あるいは空のままである。上記のことが「有向グラフ」で表現されると以下のようになる。
（１）もしノードが閉じられたループに含まれなかったなら、そして、もしいずれかの始点単語に係るノードから次々に始点単語を探索した際に当該始点単語の値が偽であるなら、ノード（単語）の値は決定されない。もし単語の値がユーザの必要条件を満たさなかったならノードの値は決定されない。たとえ値が成功裏に計算されるとしても、ノード（単語）の値が偽と決定される。
（２）もしノードが閉じられたループに含まれなかったなら、そして、もしすべての始点単語が、そのノードから次々に始点単語を探索したとき真であるなら、ノードの値は真として決定される。
（３）ノードが閉じられたループに含まれるときであって、ノードの始点単語がシステムに存在しないとき、ノード（単語）の値は空のまま残る。
（例２）
例１の場合、図３において、x_3 、x_4 、x_5 、x_6及びx_7の状態は、この場合における終端の始点単語であるx_1及びx_2の状態に依存する。
同図から明らかなように、終端の始点単語であるx_1が真になるとき、x_5は真である。終端の始点単語であるx_1及びx_2が真であるとき、x_4及びx_5が真になる。x_6及びx_7はループにあるので空のままである。

Ｉ－２－（４）隣接行列Ｆと状態ベクトルＸの計算操作
ｎ個の単語の状態はベクトルX（列１つからなる行列）に、単語の順序に従って記録される。隣接行列Fと状態ベクトルXを用いて繰り返し処理を実行して計算した状態と同じ状態を表すように、算術演算子が定義される。言い換えれば、FとXの乗算は以下のようになる。
【００６２】

【数８】

$X_m = FX_{m-1}$
この時、
X_m, X_{m-1}：　n個の単語の列ベクトル
X_0：　単語の初期ベクトルの　全ての要素は空　(null)
x_{mi} of X_m：　(−1) or (+1) or (null)
f_{ij} of F：　0 or 1
n：　単語の個数　items of words
m：　繰り返しの回数

ｍ回目の隣接行列Ｆと状態ベクトルＸの積である、状態ベクトルX_mの要素$x_{m,i}$（ｍ回目の繰り返し処理後の単語x_i）は以下のような式で表せる。

【００６３】

【数９】

$$x_{mi}=f_{i1}\cdot x_{m,1}+\cdot\cdot+f_{i(i-1)}\cdot x_{m(i-1)}+f_{ii}\cdot x_{m-1,i}+f_{i(i+1)}\cdot x_{m-1,i+1}+\cdot\cdot+f_{in}\cdot x_{m-1,n}$$

乗算演算子（・）の左側の$f_{I\,j}$には、隣接行列Fのi行目の終点単語においてｊ列の始点単語Ｘｊが使われているか否かを表す指数１か０（始点単語となる場合が１、始点単語とならない場合が０）が与えられる。乗算演算子（・）の右側の$x_{m,\ i}$には、ｍ回目の繰返し処理中のベクトルＸのｉ行目に表された始点単語x_iの状態を示す３種類の指数＋１（真）、－１（偽）、null（ここでnullは空を意味する）が与えられる。

【００６４】

数式６の隣接行列Fと数式７のベクトルXを例にして、終点単語x_3の３回目の繰返し処理を数式９で表すと以下のようになる。

$x_{3,3}=f_{3,1}\cdot x_{3,1}+f_{3,2}\cdot x_{3,2}+f_{3,3}\cdot x_{2,3}+f_{3,4}\cdot x_{2,4}$
$+f_{3,5}\cdot x_{2,5}+f_{3,6}\cdot x_{2,6}+f_{3,7}\cdot x_{2,7}$
$x_{3,3}=0\cdot(+1)+0\cdot(+1)+0\cdot(\mathrm{null})$
$+1\cdot(+1)+1\cdot(+1)+0\cdot(\mathrm{null})+0\cdot(\mathrm{null})$

数式６のy_3行と始点単語x_1列との交点は「0」、数式５で示されたx_1の状態を表す列の第１行は「+1」、よって、$f_{3,1}\cdot x_{3,1}$は「0・（＋1）」となっている。数式６のy_3行と始点単語列との交点総てについて、その指数と数式７に示された対応する状態指数を乗算演算子で結び、付加演算を用いて列記していくと上記の式となる。

ここで、乗算演算子（・）の演算結果を次のように定義する。

【００６５】

【数１０】

（公理１）

$0\cdot(-1)=(+1)$　　$0\cdot(\mathrm{null})=(+1)$　　$0\cdot(+1)=(+1)$

$1\cdot(-1)=(-1)$　　$1\cdot(\mathrm{null})=(\mathrm{null})$　　$1\cdot(+1)=(+1)$

数式１０の公理は以下のことを示している。始点単語指数が０であれば、始点単語の状態が何であれ、始点単語の状態は真（＋１）となり、終点単語の状態に影響を与えない。始点単語指数が１の場合は、始点単語の状態はそのまま変化せず、始点単語の状態が終点単語の状態に営業を与える。

【００６６】

次に、付加演算子（＋）の演算結果を次のように定義する。付加演算子に用いる値は、公理１の演算結果である。

【００６７】

【数１１】

(公理２)

(-1) + (-1) = (-1)　(-1) + (null) = (-1)　(-1) + (＋1) = (-1)

(null) + (-1) = (-1) (null) + (null) = (null)

(null) + (＋1) = (null)

(＋1) + (-1) = (-1)　(＋1) + (null) =(null)

(＋1) + (＋1) = (＋1) または (-1)

上記の公理２は、さらに右辺によって類別してみると、

【００６８】

【数１２】

＜第１群：右辺が（＋１）となる場合＞

(＋1) + (＋1) = (＋1)[右辺の値はユーザ要件を満たす]

＜第２群：右辺が（-1）となる場合＞

(-1) + (-1) = (-1)　(-1) + null = (-1)　(-1) + (1) = (-1)

null + (-1) = (-1)　(1) + (-1) = (-1)

(＋1) + (＋1) = (-1)[右辺の値はユーザ要件を満たさない]

＜第３群：右辺が（null）となる場合＞

(null) + (null) = (null)　(null) + (＋1) = (null)　(＋1) + (null) = (null)

となる。

【００６９】

数式１１乃至１２から明らかなように、付加演算子(+)の意味については、例えばもし、左辺に（－１）が１個でもあれば右辺は（－１）となる。もし左辺に（－１）がなく、かつ空(
null)が１個でもあれば、右辺は空である。 もし左辺のすべてが(＋１)であるなら、右辺の状態は、終点である右辺の値がユーザ要件を満たす場合は（＋１）、ユーザ要件を満たさない場合は（－１）である
。

公理１及び２を使うことによって、次の定理が証明される。

（定理１）

終点単語の状態が(＋１)（真）になるのは、始点単語の総ての状態が(＋１)（真）であり、かつ終点単語の値がユーザ要件を満たす場合である＜公理２第1群＞。

(証明)

もし単語の状態が（＋１）（真）であるなら、それは公理１によって、その単語が始点単語として用いられるとき、１・（＋１）＝(+1)であるから始点単語は（＋１）（真）

になる。その単語が始点単語として用いられない場合については、それは０・＊　＝（＋１）（＊は３つの状態のいずれかを表す）であるから、単語の状態がいずれであっても、公理２に用いる始点は（＋１）（真）となる。付加演算が実行された結果が（＋１）（真）になるのは、公理２によって、定理１の場合以外には考えられない。

（定理２）

終点単語の状態が（－１）（偽）になるのは、始点単語の状態が１つでも（－１）であるときか、あるいはすべての始点単語が（＋１）（真）であってかつ終点単語の値がユーザ要件を満たさない場合である＜公理２第２群＞。

（証明）

もしいずれかの始点単語として用いられる単語の状態が（－１）であるなら、公理１によって１・（－１）＝（－１）であるから、始点単語の状態は（－１）（偽）になる。そのために、結果が公理２によって（－１）（偽）
になる。すべての始点単語が（＋１）（真）であるときは、定理１及び＜公理２第２群＞によって、（－１）（偽）になると証明される。
（定理３）
　終点単語の状態が（未決定である）空になるのは、1個以上の始点単語の状態が空であり、かつ始点単語の残りの状態が（＋１）（真）である場合である＜公理２第３群＞。

（証明）

もし１個以上の始点単語の状態が空であって、残りの始点単語の総てが（＋１）（真）あるなら、公理２によって終点単語の状態は空になる。

一巡処理は$X_m = F X_{m-1}$によって表せる。繰り返し処理が継続されると以下のようになる。

【００７０】
【数１３】

$$X_1 = FX_0$$

$$X_2 = FX_1 = F^2 X_0$$

$$\vdots$$

$$X_m = FX_{m-1} = F^m X_0$$

$$X_{m+1} = FX_m = F^{m+1} X_0$$

F^mは通常のマトリックスの積として定義される。F^mの要素（i,j）はマイナスでない整数である。
　終点単語の状態は一巡処理を繰り返す繰り返し処理によって空から（＋1）　または（－1）に変化する。
その状態については、たとえ（＋１）（真）のものと（－1）（偽）のものとが繰り返し

プロセスを実行するとしても、終点の状態は変化しない。有向グラフによって表現した際、グラフに閉じられたループが存在し、終点が閉じられたループ内にあるとき、あるいは、このシステムが始点を持っていないとき、何度もそれを繰り返すとしても、終点単語の状態は空のままである。「状態に変化がない（すなわち、固定点fixed pointに達した）」ことは$X_{m+1}=X_m$で表わされ、数式８を用いると、次のようになる。

【００７１】

【数１４】

$$F^{m+1}X_0 = F^m X_0$$

この時

F：　隣接行列

X：　状態ベクトル（単語の　　状態を列で示す）

X_0：　初期状態、すなわち全　　ての要素が　null

m：　繰り返しの数

この方程式が有効なとき、プログラムは停止する。

Ｉ－３．不要な繰り返しを回避する方法（実行順序の並び替え）

Ｉ－３－（１）隣接行列 の性質

Ｉ－２で、数学的構造モデルが確立された。ユーザ要件のすべての特性が隣接行列のみで示される。したがって、この隣接行列の特性を調べるのは非常に重要であって、有用である。

【００７２】

隣接行列Fの特性を利用して、不要な反復を避けるための方法を示す。どんなプログラム(アルゴリズム)も、再帰的な方法のみを用いて説明することができる。手続的には、再帰はwhileループ（繰り返しの終了条件をwhileで指定したループ）によって実現される。

【００７３】

数式８を使用して、「いかなる状態変化もない」(すなわち、「定点に達する」)という状態は、以下のように示される:

【００７４】

【数１５】

$$F^{m+1}X_0 = F^m X_0$$

この時

F：隣接行列

X:状態ベクトル単語の状　態を列で示す

X_0:初期の状態。言い換え　れば 全ての要素が空

m:繰り返し処理の回数

この公式が有効になるとき、プログラムは止まることになる。

【００７５】

このとき、行列Fには、どんな特性があるか?行列Fmの特性は何であるか?これらの疑問に答えるために、定理４を導入することが必要である。

【００７６】

空でない有限集合Vの組G=(V、E)とV×　Vの部分集合Eは有向グラフと呼ばれる。Vの要素は頂点もしくはポイント（点）と呼ばれ、Eの要素(u,v)は縁もしくは矢と呼ばれる。Gのポイントの有限系列P、$P=<v_1, v_2, \cdots, v_n>$は$v_0$から$v_1$に至るパスと呼ばれ、各i(1≦I≦n

)について、(v_{i-1},
v_i)∈Eが成り立つときには、nはこのパスの長さと呼ばれる。v_0はPの始点、v_nは終点と呼ばれる。＜v_0＞は長さゼロのパスである。パスは隣接している矢の方向と共にGのポイントを追従する道筋である。
（定理４）
有向グラフVの隣接行列をAと定義する。A^nの(i、j)要素は、互いに異なるGのv_iからv_jまでの長さnのパスの数である。

m=１、m=２、…と連続して計算し$F^{m+1}X_0=F^mX_0$が成立するときにはプログラムが停止する。FとF^mのプロパティを検証する。X_0の全要素が常にヌル（空）なので、F
はシステムの全プロパティを表す。
１）$F^m=0$である場合　(m≦n)
F^mのすべての要素が0である。定理４を用いれば、閉鎖ループがこの隣接行列の有向グラフに存在しないことが証明される。変数mはその有向グラフの最も長いパスの長さを表す。Fの固有値はすべて0である。零べき行列(F^m
=0)のプロパティから、対角に要素0を持つ下三角形のマトリクスFに再配列することができる。即ち、
$F'=P^{-1}FP$、ここでPは正方行列もしくは長方形行列。
理由は、$F(F^mX_0)=F^mX_0$, $F^{m+1}X_0=F^mX_0$
が有効であるからである。Xmのすべての要素は(+1)か(-1)である。

２）$F^m\neq0$である場合　(m≦n)
もし、閉じたループがこの隣接行列の有向グラフに存在した場合、対角の要素
【００７７】
【数１６】

$$f_{ii}^{(r)} : i = 1, \cdots, n \ \text{ of } F^r$$

に１つ以上の整数が現れることが、定理４を使って証明される。このとき、変数ｒは最も短い閉じたループである。Fの固有値は、総てが０ではない。言い換えれば、固有値の１つは０と等しくない。総てのX_0値が空（null）なので、$F^{m+1}X_0=F^mX_0$は有効である。変数ｍは、その有向グラフのループの前の最も長いパスの長さを表す。X_mの例では、
【００７８】
【数１７】

$$X_m = {}^t[(+1),(+1),(\text{null}),(\text{null}),(\text{null}),(\text{null})]\ .$$

もし、閉じたループが有向グラフに存在しなければ、その隣接行列は零冪行列（$F^m=0$）となる。この特性は重要である。

Ｉ－３－（２）繰り返しプロセスを不必要にする単語の並べ方

「有向グラフ」で表現する際に、前述の第III章１項で示したように、閉じられたループ内で始点単語・終点単語である関係がないときには、隣接行列が下三角行列になり、対角の三角形が０になるような一巡処理順序に単語を並べることによって、繰り返しプロセスを避けることができる。
【００７９】
分析的に規則正しい行列Pを得るのは困難である。$F'=P^{-1}FP$を用いる代わりに、トポロジカル・ソートを用いる。半順序関係が与えられているとき、最初にするべき仕事から

終わりにするべき仕事まで、データを一列に並べることをトポロジカル・ソート（topological sort）と呼ぶ。半順序関係（partial order）であるから、答えが常に唯一的に決定されるわけではない。

【００８０】

有向グラフに深さ優先探索（depth first search：縦型検索ともいう）をしつつ、戻るときにノードを拾い上げるという方法によって、トポロジカル・ソートを実行することができる。各ノードからこれを行い結果を逆順にならべる。順序処理(一巡処理）における単語を並べるには、この方法によって単語を並べさえすればよい。

図４は、あるノード１乃至８が形成する特定の順序関係を表した有向グラフである。

【００８１】

即ち、同図に示す例においては、先に定義した始点単語と終点単語との関係を用いて次のように規定できる。

ノード２≦ノード１
ノード１≦ノード３
ノード３≦ノード４、ノード５≦ノード４
ノード２≦ノード５
ノード５≦ノード６
ノード３≦ノード７、ノード６≦ノード７、ノード８≦ノード７
ノード６≦ノード８

（例４）

図４に示される有向グラフの隣接行列は次の数式１８の通りである。

【００８２】

【数１８】

$$F=\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array},\quad X=\begin{bmatrix} \text{null} \\ 1 \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \end{bmatrix}$$

図４の有向グラフでは閉じられたループ内で始点単語・終点単語の関係がない。そこで、この半順序関係に係る隣接行列が下三角行列になるように、トポロジカル・ソートを行う。具体的には次のように行う。

有向グラフに対する深さ優先探索を実行する。即ち、一例として、ノード１を始点とすると、ノード１を出発して連結されている未訪問のノードを半順序関係を守って探索（即ち

、連結される未訪問のノードまで進み、行き場所がなくなったら、行ける場所があるノードまで戻って（再帰）再び進めるだけ進み、行き場所がなくなったら終了（再帰からリターン）：これを「再帰的に進む」ともいう）する。これにより、ノード１→ノード３→ノード４（＊）→ノード３→ノード７（＊）→ノード３（＊）→ノード１（＊）を得る。但し、（＊）は再帰の時点を示す。
次に、始点をノード２に移し、上記と同様の探索を行う。すでに訪問したところは進まない。これにより、ノード２→ノード５→ノード６→ノード８（＊）→ノード６（＊）→ノード５（＊）→ノード２（＊）を得る。
その他のノードを始点として探索をするところ、すでに全ノードを訪問済みなので探索は終了する。
戻る（再帰の）ときのノードを拾い上げる。具体的には、（１）（２）で（＊）の時点のノードを拾い上げる。上の例では、ノード４（＊）→ノード７（＊）→ノード３（＊）→ノード１（＊）→ノード８（＊）→ノード６（＊）→ノード５（＊）→ノード２（＊）を得る。
順序処理(一巡処理）における単語を逆順に並べる。即ち、（４）で拾い上げたノードの列を逆に並べたものを一つの解として得る。上の例では、ノード２→ノード５→ノード６→ノード８→ノード１→ノード３→ノード７→ノード４が求める一つの解である。

以上より、トポロジカル・ソートの結果は（１，２，３，４，５，６，７，８）→（２，５，６，８，１，３，７，４）である。（「ノード」の表記は省略する。）

このトポロジカル・ソートの具体的な方法についてフローチャートを用いて説明すると次のようになる。図５はトポロジカル・ソートの動作を説明するためのフローチャートである。

【００８３】
まず、初期値を、a=1,I=1と設定する（ステップ５０１）。

【００８４】
次に、a=a+1とし、始点をノードaに設定する（ステップ５０２）。

【００８５】
ここから探索を開始する。具体的には、訪問可能な連結されているノードで未訪問のところがないかを判定する（ステップ５０３）。これについては、例えば、各ノードに訪問フラグを設定、初期値に０（未訪問を示す）を入れておき、当該ノードが訪問されたときには訪問フラグを１にするなどの方法により自動判定させることができる。

【００８６】
ここで未訪問ノードが存在するときには、１ノード進み（ステップ５０４）、ステップ５０３に戻る。

【００８７】
未訪問ノードが存在しないときには、１ノード戻り（再帰）、Ｉ＝Ｉ＋１とする（ステップ５０５）。このとき、再帰に係るノードを再帰ノード列Ｉ番目の要素として退避する（ステップ５０６）。

【００８８】
次に、再帰の結果ノードaに戻ったかを判定し（ステップ５０７）、まだ戻っていなければステップ５０３に戻る。ノードaに戻った場合には始点をノードaとする探索は終了する。即ち、再帰からリターンし、Ｉ＝Ｉ＋１とし（ステップ５０８）、再帰点を先の再帰ノード列Ｉ番目の要素として退避する（ステップ５０９）。

【００８９】
ここで全ノードが探索済みかどうかをチェックする（ステップ５１０）。まだ探索を行っていないノードが存在する場合には、始点を替えて再度探索を始めるため、ステップ５０２に戻る。全ノードの探索が終了しているときには、探索は終了し、退避してあった再帰ノード列を逆順に並べる（ステップ５１１）ことで、並び替えの１候補を得て、一連の

トポロジカル・ソートの動作を終了する（ステップ５１２）。

【００９０】

このように、トポロジカル・ソートのアルゴリズム自体を上記で説明したフローチャートに沿ってプログラム化すれば、トポロジカル・ソートの自動プログラムを得ることができる。

【００９１】

さて、上記によりＦをトポロジカル・ソートした。その結果、Ｆ’（トポロジカル・ソートされたＦ）は次の数式１９のようになる。

【００９２】

【数１９】

$$F'=\begin{array}{c} \\ 2 \\ 5 \\ 6 \\ 8 \\ 1 \\ 3 \\ 7 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccccccc} 2 & 5 & 6 & 8 & 1 & 3 & 7 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

トポロジカル・ソートの結果、一巡処理の順序は、「1，2，3，4，5，6，7，8」から「2，5，6，8，1，3，7，4」となった。

【００９３】

このようにＦ’が得られるが、このＦ’はその形から明らかなように上三角形に当たる総ての項が０である下三角行列になっている。この（ソート済み隣接行列である）Ｆ’は（上三角形部分には０しかないため）その構造上、あるノード（終点単語）を求めるための始点単語は当該ノードより（ソート済み隣接行列における）上の行のノードにしかない。従って、Ｆ’の行の上位から順番に値を求めていけば、Ｆで同じことを行う場合に比べて、繰り返しや遡りを行うことなくスムーズに総てのノードについて値が求まる。

【００９４】

さて、次に、上記で説明したトポロジカル・ソートをＬｙｅｅ（登録商標）方法論において応用する方法の一つを説明する。

【００９５】

図６は、Ｌｙｅｅ（登録商標）方法論における単語単位の論理要素をノードとして上記のトポロジカル・ソートを用いて繰り返しをなくすための自動制御の動作を説明するため

のフローチャートである。

【００９６】

同図に示すように、まず、単語ごとの論理要素（これらをノードと考える）間の関係を規定する（ステップ６０１）。

【００９７】

このためには例えば有向グラフの作成（ステップ６０２）を行っても良いが、これに限定されるものではない。

【００９８】

次に、ノードについての（上記説明した）隣接行列を定義する（ステップ６０３）。

【００９９】

この隣接行列に対して上述のトポロジカル・ソートを実行する（ステップ６０４）。トポロジカル・ソートの詳しいアルゴリズムについては図５及び対応する説明で詳述したので、ここでは割愛する。また、ここでは探索技法の一例としてトポロジカル・ソートとして説明したが、このステップ６０４では前述の他の探索の実行を行っても良いのはもとよりである。

【０１００】

最後に、トポロジカル・ソートの結果得られた１解に基づき、当該単語単位の論理要素（ノード）を並び替え（ステップ６０５）、当該動作を終了する（ステップ６０６）。

【０１０１】

なお、隣接行列定義のステップ（ステップ６０３）については、本稿では理解の容易さを助けるために入れたが、本質的には必須のステップではない。つまり、隣接行列を定義することなく、単語ごとの論理要素間関係の規定（ステップ６０１）から、直接に、或いは有向グラフ作成（ステップ６０２）を介して、トポロジカル・ソート等の探索のステップ（ステップ６０４）に至ることも本発明の思想の範囲内であり、本発明の目的とするところを実現し得る。

【０１０２】

このように、Ｌｙｅｅ（登録商標）のソフトウェア生成自動プログラムにかける前の段階で、Ｌｙｅｅ（登録商標）方法論によるユーザ要件から抽出した単語ごとの論理要素に対してトポロジカル・ソートをかけることで繰返しの省かれたＬｙｅｅ（登録商標）方法論によるソフトウェア生成が可能となる。

【０１０３】

換言すれば、上記のノードをＬｙｅｅ（登録商標）方法論における単語ごとの論理要素と考えれば、上記の方法でＦ’を求めることで、ある単語について値を求める（有意性を決定する）に当たって、何回も繰り返し処理を行うことなく１回のＲＵＮのみで目的を達することが可能となる。

【０１０４】

さらに、上記のようなトポロジカル・ソートのアルゴリズムは、上記の説明から明らかなように、それ自体をソフトウェアとしてプログラミングして自動化することが可能である。

【０１０５】

従って、Ｌｙｅｅ（登録商標）方法論によるソフトウェア自動生成方法についてそれをさらに効率化・高品質化するのにも自動化を図れることになる。

【０１０６】

なお、上記の説明においては、有向グラフに表される単語の順序関係として主として半順序関係を持つ場合について説明したが、半順序関係のみならず、全順序関係について、或いは全順序関係及び半順序関係が混在する場合についても上記の説明が該当することはもとよりである。

【０１０７】

また上記の説明においては、理解のしやすさの観点から、有向グラフの作成を行った後に隣接行列を作成する点を説明したが、有向グラフの作成を介さずに、単語単位のプログ

ラム相互の関係から直接隣接行列を作成するようにしても本発明は実現できる。

すべての単語で作るべきプログラムの構造と当該プログラムを制御するのに用いるパレット連鎖関数のプログラムの構造は、上記の記述で明確になった。そして、それらが数学上適切に動くことが証明された。

Ｉ－４．まとめ

以上詳細に説明したように、トポロジカルソート技術では、単語の状態を定義している単語単位をベースにするプログラムの数学的な構造モデルを作成し、それをＬｙｅｅ（登録商標）方法論によるソフトウェア開発に適用することで、より高効率、高速、高パフォーマンスの開発を実現した。

【０１０８】

単語の状態は完全な半順序を構成する。それは単語の最小要素（極小点）として 空を持っている。有向グラフとその隣接行列とを使って、この構造モデルは数学的な面からも正しいといえる。

【０１０９】

さらに、本技術ではパレットの一般的な考えと基本構造の正当性を考察した。それらは実務上、プログラムを作るために必要である。また、順次読込（sequential read)のようなアルゴリズム、合計値算出を実現するための構造について説明した。これらの一般的な考えは Lyee（登録商標）のシステム開発方法論で使われる。それは多くの産業的な応用分野ですでに実用的に有効とされた多くの結果をもたらしている。

人間であるユーザと、外の装置とのインタフェース及びコミュニケーションについても論じられている。これは、このようなシステムのタイプのために、単語単位をベースとするプログラムとして設計する（モデリング）ことの、実際の作業を正当化する。

以上を要するに、本技術によれば、数学的な考え方、手法、解析法をＬｙｅｅ（登録商標）のソフトウェア開発論、開発技術に適用・応用するので、ソフトウェア開発をより効率良く、高速に、精密に実現することができる。

【０１１０】

上記の詳細な説明では、有向グラフ表記、隣接行列算出、トポロジカル・ソート、並び替えという順序で本発明の思想を実現する方法について述べた。しかしこれらはどれも絶対不可欠な要素というものではなく、例えば、有向グラフ表記の表記を経ないで直接隣接行列算出→トポロジカル・ソート→並び替えという順序で本発明の思想を実現してもよい。

【０１１１】

さらに、有向グラフに対して、例えば幅優先探索（Ｂｒｅａｄｔｈ－Ｆｉｒｓｔ Ｓｅａｒｃｈ）、反復深化法（Ｉｔｅｒａｔｉｖｅ Ｄｅｅｐｉｎｇ）、発見的探索（Ｈｅｕｒｉｓｔｉｃ Ｓｅａｒｃｈ）、ヒル・クライミング（Ｈｉｌｌ Ｃｌｉｍｂｉｎｇ）、最適優先探索（Ｂｅｓｔ－Ｆｉｒｓｔ Ｓｅａｒｃｈ）、オイラー（Ｅｕｌｅｒ）の一筆書き、ダイクストラ（Ｄｉｊｋｓｔｒａ）法等の各種探索技法を組み合わせることで並び替えてもよい。

【０１１２】

或いは、隣接行列算出のあとを、これらの各種探索技法を組み合わせることで本技術の根本思想を実現してもよい。

【０１１３】

図７は、本技術をソフトウェア開発装置として実現する場合の機能構成を示したブロック図である。即ち、同図に示す一例においては、ソフトウェア開発装置１０００として、

次の要素を備えさせる。

【０１１４】

制御部１０１０は全体の制御及びＬｙｅｅ（登録商標）方法論に基づいたソフトウェア開発を自動的に行う機能を有する。即ち、後述する（有向グラフ作成部１０５０、）隣接行列計算部１０６０及びトポロジカルソート部１０７０により並び替えられた単語単位プログラム群をＬｙｅｅ方法論に自動的に適用して生成コードを得る機能を有している。

【０１１５】

情報入力部１０２０は（図示しない）画面等を通じて、Ｌｙｅｅ(登録商標)に係るユーザ要件を初めとする人間の意思をコンピュータに入力するインターフェースとしての機能を果たす。情報取得部／通信部１０３０は外部との通信や（ソフトウェア開発装置１０００にある場合の）データベース１０４０、１０４１等との情報の授受を行う。なお、この例ではデータベース１０４０、１０４１はソフトウェア開発装置１０００内に持っていてもよいことはもとよりである。

【０１１６】

有向グラフ作成部１０５０は、制御部１０１０の制御に基づき、上記で説明したように、Ｌｙｅｅ(登録商標)に係るユーザ要件から半順序関係を割り出し、それを有向グラフとして作成する機能を有する。隣接行列計算部１０６０は、制御部１０１０の制御に基づき、有向グラフ作成部１０５０で作成された有向グラフについて、上記で説明したような隣接行列を計算・作成する。トポロジカルソート部１０７０は、制御部１０１０の制御に基づき、隣接行列計算部１０６０で計算・作成された隣接行列に対して、上記で説明したようなトポロジカルソートを実行する機能を有する。

【０１１７】

なお、上記トポロジカルソート部１０７０は探索を行う機能を持つ一例であって、上述のように、他の探索手法を実現する機能を有する探索部（一般）としても本発明の思想は実現される。

【０１１８】

このように構成することにより、これら各部（１０１０～１０７０、従って１０００）は総て自動化できるので、結局繰返し処理の排除→ソフトウェア生成まで一貫して自動処理が可能となり、ソフトウェア生産速度、効率は各段に向上する。

【０１１９】

このように、各機能を、それを実現するようにプログラムとして組んだソフトウェアとして、或いはそのようなソフトウェアを例えばＲＯＭ（ＲｅａｄＯｎｌｙＭｅｍｏｒｙ）に書きこみかかるＲＯＭを組み込んだ全体システム（装置）とすることで、前述した本発明に係る各要素をソフトウェアとして或いはそのようなソフトウェアが組み込まれたソフトウェア開発装置としても実現可能である。

ＩＩ章　Ｌｙｅｅ方法論のシステム構造の単一化

本章では、Ｉ章で述べた単語単位のプログラムの最適順序への並び替えによるプログラムの効率化技術を、システム全体に適用してシステム全体の効率化を実現するための、Ｌｙｅｅ方法論のシステム構造の単一化技術について述べる。

ＩＩ－１　Ｌｙｅｅ方法論のシステムの構造

ＩＩ－１－（１）Ｌｙｅｅ方法論のシステムを構成する単位：基本構造

Ｌｙｅｅ方法論は、同一条件を持っている単語を１つのグループとして、システム全体をそのグループ単位に分割する。同一条件とは、同時に出力（画面またはファイルへ）される、という条件で、そのグループの単位を論理レコードと呼ぶ。論理レコード単位に単語単位のプログラムが基本構造（Basic Structure）と呼ぶ構造を作り、基本構造単位に単語単位のプログラムが実行される。これは Ｌｙｅｅの規則の１つである。

【０１２０】

従って、Ｌｙｅｅ方法論においては、１つのシステムはいくつかの基本構造に分割され

ることになる。システムのこの構造は、処理経路図（Process Route Diagram。あるいはPRDと呼ぶ）によって表現される。処理経路図では、基本構造のリンクの状態（実行の経路を示す）が表現されている。図８は２つの基本構図からなるシステムの場合の処理経路図の例である。

【０１２１】

基本構造から別の基本構造へとプログラムの実行が遷移するしくみは、図８の例で説明すると次のとおりである。第１の基本構造において、１つ以上の始点単語（start-word）の状態がnull（未決定）　であることによって終点の値の状態がnull（未決定）である出力単語が１つ以上あるとき、次に実行されるべき基本構造（第２の基本構造）が深さ優先探索（depth-first
search）によって探索され、次に実行する基本構造として指定される。次に実行される第２の基本構造において、値がnullである出力単語がなくなったとき、第２の基本構造は実行を終了し、次に実行する基本構造として、null値が最初に現われた第１の基本構造が指定される。このプロセスは、システム全体の単語の状態変化が総て消滅するまで継続される。

このように、Ｌｙｅｅ方法論によるシステムは、基本構造単位の構造を持つので、システムの実行は基本構造単位の実行順序となる。この場合、基本構造単位の実行は１巡処理で完了することは保証されていおらず、単語単位のプログラムの実行は無駄な繰り返しが発生する。

【０１２２】

これに対して、システムとして実行時の構造として、システム全体を１つの構造に統合すれば、基本構造単位での順序ではなく、システム内の単語単位プログラム全部に対して、一括にＩ章で述べたトポロジカルソートを施すことができ、単語単位プログラムを最適順序で実行することによってプログラムの実行時間等の効率化がはかれる。

基本構造から次に実行すべき基本構造の決定が、深さ優先探索では決定できない場合がある。それは、ユーザの画面からの指示によって実行される基本構造が決定する場合である。たとえば、画面に処理Ａを実行するボタンと、処理Ｂを実行するボタンがあり、ユーザがこれらのボタンのいずれか１つを押下することによって、プログラムが実行する処理内容が決定するような場合である。このように、画面からのユーザの指示によって処理内容が決定するような仕様を含むシステムにおいては、基本構造単位に分割され単語単位プログラムの実行条件をそのままにして、基本構造を１つの構造に統合し、トポロジカルソートを施すことができない。以下に、画面からのユーザ指示が実行する基本構造を決定するような場合の基本構造の単一化の方法について具体例を上げて以下に詳しく述べる。

そのためのステップとして、まず、基本構造をリンクする（次に実行すべき基本構造を指定する）役割を果たしているプログラム、経路作用要素について詳しく考察する。

Ｉ-Ｉ－１－（２）基本構造をリンクするプログラム：経路作用要素

次に実行する基本構造を指定することによって基本構造をリンクする役割は、経路作用要素（Routing Vector）と呼ぶプログラムが行う。Ｌｙｅｅの単語単位プログラムにおいては、入力単語、状態単語の値を生成するための単語のプログラム以外に、次に実行する基本構造を指定する、あるいは出力単語の値を出力する、などの作用を行うためのプログラムがある。これらのプログラムを作用要素（action
vector）と呼び、経路作用要素もその１つである。

これらの作用要素も一種の単語と見なすことができ、実際これらのプログラムの基本的構造は単語のプログラムと同じである。単語と作用要素の大きな違いは、単語は１つの単

語のみに作用（値の生成という作用）するが、作用要素は同時に２つ以上の単語に対して作用する点だけである。たとえば、経路作用要素（次に実行する基本構造を指定する）は、基本構造という単語の集合に対して作用し、出力作用要素（出力単語を出力する）は、出力レコードという単語の集合に対して作用する。作用要素を単語とみなすことによって、作用要素を有向グラフおよび隣接行列の要素として取扱うことができ、システム全体をＩ章で説明してきた数学的モデルによって表すことができる。

【０１２３】

以下に例を上げて、単語単位プログラムのモデルにおける経路作用要素の意味を説明する。

（例５）

図９は、あるシステムの画面で、ユーザがデータ項目ｃおよびデータ項目ｄにデータを入力して実行ボタンのａまたはｂを押すことによって、データ項目ｇに値を得るためのシステムである。７０１はデータ項目ｃの入力データフィールド、７０２はデータ項目ｄの入力データフィールド、７０３はデータ項目ｇの出力データフィールド、７０４は実行ボタンａ、７０５は実行ボタンｂである。実行ボタンａが押されるときは、ｇの値は式ｃ＋ｄで計算され、実行ボタンｂが押されるときは、ｇの値は式ｃ×ｄによって計算される。実行ボタンのａおよびｂは、両方が同時に押されることはないので、ｇの値の生成のための２つの計算式のどちらが実行されるかの条件となっている。

【０１２４】

このシステムを基本構造単位に単語で表すと図１０のようになる。図１０のＢＳ１、ＢＳ２、ＢＳ３はそれぞれ基本構造である。基本構造ＢＳ１は、図９に示した画面の基本構造である。ＢＳ１は、実行ボタンである入力単語ａと入力単語ｂ、入力データ項目である入力単語ｂと入力単語ｃ、出力データ項目である出力単語ｇ、経路作用要素であるX_rを含んでいる。

【０１２５】

基本構造ＢＳ２は、実行ボタンａが押された時、出力データ項目ｇの値を式ｃ＋ｄによって計算するための基本構造（媒体がファイル）である。式ｃ＋ｄの結果を記録する単語ｅを含んでいる。基本構造ＢＳ３は実行ボタンｂが押された時、出力データ項目ｇの値を式ｃ×ｄによって計算するための基本構造（媒体がファイル）である。式ｃ×ｄの計算結果を記録する単語ｆを含んでいる。

【０１２６】

画面にユーザがデータを入力した（ＢＳ１が実行された）後に、出力単語ｇの値を計算するための基本構造はＢＳ２またはＢＳ３であるが、どちらが実行されるべきかを指定するのが経路作用要素X_rである。２つの基本構造に所属する単語は、それぞれボタンａとｂのどちらが押されたかによって実行が決まるので、経路作用要素X_rは、実行ボタンａが押された場合（単語ａ＝真、値がある）はＢＳ２を、実行ボタンｂが押された場合（単語ｂ＝真、値がある）はＢＳ３を、次に実行する基本構造として指定する。また、実行が次の基本構造に移るのは入力データがそろったときであるので、単語ｃおよび単語ｄに値があることも、経路作用要素X_rの実行条件となる。

【０１２７】

基本構造ＢＳ２またはＢＳ３の実行によって、出力単語ｇの値が単語ｅまたは単語ｆに成立する。出力データを画面に出力するための基本構造ＢＳ１の出力単語ｇは、単語ｅに値が成立したとき（ｅ＝真）はｇ＝ｅ、単語ｆに値が成立したとき（ｆ＝真）はｇ＝ｆ、と定義される。

【０１２８】

上記のような単語の定義を整理すると下記の表２のとおりである。

【０１２９】

【表２】

単語	入出力属性	定義式	定義式実行条件
a	入力	－	－
b	入力	－	－
c	入力	－	－
d	入力	－	－
x_r	（経路作用要素）	ＢＳ２を指定	（１）ａ＝真（ボタンａが押され、ａに値がある）かつ、ｃ＝真およびｄ＝真
		ＢＳ３を指定	（２）ｂ＝真（ボタンｂが押され、ｂに値がある）かつ、ｃ＝真およびｄ＝真
e	出力	ｃ＋ｄ	X_rがＢＳ２を指定
f	出力	ｄ×ｄ	X_rがＢＳ３を指定
g	出力	e	ｅ＝真（ｅに値がある）
		f	ｆ＝真（ｆに値がある）

ＩＩ－２　基本構造の統合によるシステム構造の単一化

ＩＩ－２－（１）　基本構造の隣接行列

図１１は、図１０の単語の関係を有向グラフによって示した図である。基本構造ＢＳ１に属する入力単語である単語a、単語ｂ、単語ｃ、単語ｄは、有向グラフによって表される単語のネットワークの最端始点となる。同じく基本構造ＢＳ１に属する経路作用要素である単語X_rは、上記表２に示したとおり、値が成立（次に実行する基本構造決定の真偽判定が成立）するためには、単語aまたは単語ｂと単語ｃと単語ｄが必要であるので、これらの４つの単語を始点とする矢印によって４つの単語とリンクされる。基本構造ＢＳ２に属す単語ｅ、および、基本構造ＢＳ３に属す単語fは、値が成立するために、単語c、単語dおよび単語X_rが必要であるので、これらの３つの単語を始点とする矢印によって各々３つの単語とリンクされる。基本構造ＢＳ１に属す出力単語ｇは、値が成立するために、単語ｅまたは中間単語ｆが必要であるので、これらの２つの単語を始点とする矢印によって２つの単語とリンクされる。

【０１３０】

次に、図１１で有向グラフで示した例５のシステムを隣接行列を用いて表す。最初に図１１の、入力単語を扱った基本構造ＢＳ１（９０１）を表した隣接行列Ｆ１を下記の数式２０に示す。

【０１３１】

基本構造ＢＳ１（入力単語）の隣接行列

【数２０】

$$FI = \begin{array}{c} \\ a \\ b \\ c \\ d \\ x_r \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & x_r \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

ＢＳ１（入力単語）の要素は単語ａ、単語ｂ、単語ｃ、単語ｄ、単語x_rであるから、Ｆ１の行列はこれらの要素によって構成される。入力単語である単語ａ、単語ｂ、単語ｃ、単語ｄは、始点単語を持たないので、終点単語a、ｂ、ｃ、ｄのそれぞれの横列（始点単語ａ、ｂ、ｃ、ｄ、x_rとの交点）は総て始点単語として用いないことを示す「０」となる。経路作用要素である単語x_rは、図１１の有向グラフに示したとおり、始点として単語a、単語ｂ、単語ｃ、単語ｄを持つので、終点単語x_rの横列は、これらの始点単語との交点は始点であることを示す「１」となる。

【０１３２】

次に、基本構造ＢＳ２およびＢＳ３（以下、基本構造ＢＳ２＆ＢＳ３と記す）を合わせたものを表す隣接行列Ｆ２を下記の数式２１に示す。

【０１３３】

基本構造ＢＳ２＆ＢＳ３の隣接行列

【数２１】

$$F2=\begin{array}{c} \\ e \\ f \end{array}\begin{array}{c} \begin{array}{cc} e & f \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{array}$$

ＢＳ２＆ＢＳ３の要素は単語ｅと単語ｆのみであるので、Ｆ２の行列はこれら２つの要素で構成される。単語ｅと単語ｆはお互いに独立している（すなわち、どちらの単語も始点単語としてもう一方の単語を用いていない）ので、いずれの交点も「０」となる。

【０１３４】

基本構造ＢＳ１の出力単語を扱った集合の隣接行列Ｆ３を、下記の数式２２に示す。

【０１３５】

基本構造ＢＳ１（出力単語）の隣接行列

【数２２】

$$F3=\begin{array}{c} \\ g \end{array}\begin{array}{c} \begin{array}{c} g \end{array} \\ \begin{bmatrix} 0 \end{bmatrix} \end{array}$$

ＢＳ３の要素は単語ｇのみであるので、Ｆ３の行列は単語ｇ１つのみで構成される。単語ｇは始点として自身を用いないので、交点は「０」となる。

【０１３６】

以上で、例５のシステムを構成する基本構造単位に、同一基本構造内の要素間の関係を隣接行列に表した。

Ｉ－２－（２）　結合隣接行列による基本構造の統合

次に、これらをシステム全体として１つの隣接行列として表すために、各基本構造の要素である単語と他の基本構造の要素である単語との関係を示すことによって基本構造をリンクする結合隣接行列（connection matrix）について説明する。

任意の基本構造の単語と他の基本構造の単語の関係は、図１１に示したように終点単語と始点単語の関係によって表わされている。基本構造同士は、それに属する単語の関係よってリンクされていると言える。図１１から以下のことが分かる。

１）基本構造ＢＳ１（入力）の単語を終点単語としたとき、いずれの他の基本構造の単語も、ＢＳ１（入力）の単語より後に現れるので、始点単語となりえない。

２）基本構造ＢＳ２＆３の単語を終点単語としたとき、ＢＳ２＆３の単語より前に現れるＢＳ１（入力）の単語は始点単語となりえるが、ＢＳ１（出力）の単語は始点単語となりえない。

３）基本構造ＢＳ１（出力）の単語を終点単語したとき、ＢＳ２＆３およびＢＳ１（入力）のいずれの単語も、ＢＳ１（出力）の単語より前に現れるので、始点単語となりえる。

上記の考察の結果、基本構造をリンクするための結合隣接行列として重要なのは、任意の基本構造と、それに属す単語の始点単語になりえる単語を含む基本構造との関係を表す結合隣接行列であることがわかる。なぜなら、有向グラフで示されるように、単語間の関係は始点単語から終点単語へと向かう関係のみが存在するのであるから、基本構造間の関係も、それと同様に始点単語を含む基本構造から終点単語を含む基本構造へと向かう関係のみが存在し、逆の関係は存在しない（そのような基本構造間の関係を表す結合隣接行列は、すべての交点が０となる）からである。

【０１３７】

以下に、例５の基本構造をリンクする（始点単語を含む基本構造から終点単語を含む基本構造へと向かう関係を表す）結合隣接行列を１つづつ考察する。
数式２３は、基本構造ＢＳ１（入力）（始点単語）から基本構造ＢＳ２＆ＢＳ３（終点単語）への関係を示す結合隣接行列ＦＣ１である。

【０１３８】

【数２３】

$$FC1=\begin{array}{c} \\ e \\ f \end{array}\begin{array}{c} \begin{array}{ccccc} a & b & c & d & x_r \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

隣接行列ＦＣ１は、基本構造ＢＳ２＆ＢＳ３の要素である単語ｅと単語ｆを終点単語においたとき、ＢＳ１（入力）の要素である単語ａ、単語ｂ、単語ｃ、単語ｄ、単語x_rが始点単語として用いられる状態を表し、２つの基本構造の関係を示している。図１１で示したように、単語ｅおよび単語ｆは、は単語ｃ、単語ｄ、単語x_rを始点単語に用いる。従って、ＦＣ１において、終点単語ｅおよび終点単語ｆの行と、始点単語ｃ、ｄ、x_rとの交点は「１」となり、その他の始点単語との交点は「０」となる。

【０１３９】

数式２４は、基本構造ＢＳ１（入力）および基本構造ＢＳ２＆ＢＳ３（始点単語）から、基本構造ＢＳ１（出力）（終点単語）への関係を示す隣接行列ＦＣ２を表している。

【０１４０】

【数２４】

$$FC2=\begin{array}{c} \\ g \end{array}\begin{array}{c} \begin{array}{ccccccc} a & b & c & d & x_r & e & f \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

隣接行列ＦＣ２は、基本構造ＢＳ１（出力）の要素である単語ｇを終点単語においたとき、有向グラフにそれ以前に現われるＢＳ１（入力）、ＢＳ２、ＢＳ３の要素である単語ａ、単語ｂ、単語ｃ、単語ｄ、単語x_r単語ｅ、単語ｆ、が始点単語として用いられる状態を表して、２つのグループの関係表している。図１１で示したように、単語ｇは、単語ｅと単語ｆを始点単語に用いる。従って、ＦＣ２において、終点単語ｇの行と、始点単語ｅおよびｆとの交点は「１」となり、その他の始点単語との交点は「０」となる。

【０１４１】

以上が、例５のシステムを構成する基本構造間のリンクに意味を持つ結合隣接行列であった。システム全体として１つの隣接行列へ統合する過程の説明を簡便にするために、下記に、関係が存在しない基本構造間の結合隣接行列も示す。数式２５は、基本構造ＢＳ２＆ＢＳ３（始点単語）から基本構造ＢＳ１（入力）（終点単語）の関係がないことを示す

結合隣接行列ＦＣ３である。

【０１４２】

【数２５】

$$FC3 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ x_r \end{array} \begin{array}{c} \begin{array}{cc} e & f \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{array}$$

数式２６は、基本構造ＢＳ１（出力）（始点単語）から基本構造ＢＳ１（入力）（終点単語）の関係がないことを示す結合隣接行列ＦＣ４である。

【０１４３】

【数２６】

$$FC4 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ x_r \end{array} \begin{array}{c} \begin{array}{c} g \end{array} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

数式２７は、基本構造ＢＳ１（出力）（始点単語）から基本構造ＢＳ２＆ＢＳ３（終点単語）の関係がないことを示す結合隣接行列ＦＣ５である。

【０１４４】

【数２７】

$$FC5 = \begin{array}{c} \\ e \\ f \end{array} \begin{array}{c} \begin{array}{c} g \end{array} \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{array}$$

上記に示した隣接行列Ｆ１からＦ３、結合隣接行列ＦＣ１からＦＣ５を用いて、例５のシステムを１つの隣接行列Ｆとして表したものが数式２８である。

【０１４５】

【数２８】

$$F=\begin{bmatrix} F1 & 0 & 0 \\ FC1 & F2 & 0 \\ FC2 & & F3 \end{bmatrix}=\begin{array}{c} \\ a \\ b \\ c \\ d \\ x_r \\ e \\ f \\ g \end{array}\begin{array}{c} \begin{array}{cccccccc} a & b & c & d & x_r & e & f & g \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

上記数式で、「Ｆ＝」の右辺に示された、Ｆ１、Ｆ２、Ｆ３、ＦＣ１、ＦＣ２と０（行列の交点が総て０である隣接行列で、ＦＣ３、ＦＣ４、ＦＣ５に相当する）で構成された行列は、基本構造単位の隣接行列と結合隣接行列がどのように統合されてシステム全体を表わすかを示している。次に、それらの要素を単語に変えたものが、その右辺にあるａ、ｂ、ｃ、ｄ、ｘｒ、ｅ、ｆ、ｇを行列の要素とする隣接行列である。Ｆ１、Ｆ２、Ｆ３、ＦＣ１、ＦＣ２、ＦＣ３、ＦＣ４、ＦＣ５がどのように統合されてＦを構成するのか、さらにわかりやすく示したのが図１２である。図１２の１００１の部分がＦ１、１００２がＦ２、１００３がＦ３、１００４がＦＣ１、１００５がＦＣ２である。行列の交点が総て０である結合隣接行列は、ＦＣ３が１００６、ＦＣ４が１００７、ＦＣ５が１００８を構成する。

【０１４６】

以上のようにして、２つ以上の基本構造から成る１つのシステムは、１つの隣接行列で表すことができる。

以上でシステム全体を１つの隣接行列で表わすことができたので、このシステムの隣接行列に対して、Ｉ章で述べたトポロジカルソートを行って、総ての単語の値の状態が最少実行回数で確定（すなわち総ての出力単語の生成が完了）するような最適順序に単語単位のプログラムの並び替えを行う。

【０１４７】

例５のシステムの隣接行列Ｆをみれば、図１３に示すように、値が１である始点単語との交点は総て左下三角形（１３０１）の内にある。Ｉ章で述べたように、これは単語が最適順序に並んでいることを意味する。従って、数式２８の隣接行列Ｆは、すでにトポロジカルソート済みであって、初期値の単語の状態ベクトルを与えれば、最少実行回数で出力単語の生成を完了することができる単語の順序になっているといえる。

ここで、トポロジカルソートを施す効果について補足する。トポロジカルソートによって最適実行順序に並んだ単語単位のプログラムは、１巡処理で総ての出力単語の値を生成することができる。しかし、システム全体に対して言及する場合には、より正確には「最少実行回数」で実現できるということが適切である。なぜなら、システム全体でみたときには、集計処理のように、一時的記録領域を置くことによって同じ計算式を繰り返し利用する必要がある処理を含む場合があるからである。もちろん、この場合の「繰り返し」は、無駄な繰り返しではない。集計するデータの数マイナス１が最少繰り返し数になる。こ

のような集計処理も、結合隣接行列を用いて１つの隣接行列に表わすことができる。

ＩＩ－２－（３）　システムの隣接行列の検証

本セクションでは、前セクションで示したシステムの隣接行列Ｆが、出力単語の値を生成する関数として機能し、しかも、トポロジカルソート済みであるので、最少実行回数で総ての出力単語の生成を完了できることを、Ｉ章で述べた単語の状態ベクトルをかける計算操作によって検証する。出力単語ｇの値が決定されれば、隣接行列Ｆは関数として機能すると言える。また、例５のシステムは、前述の集計処理のような繰り返し処理を含まないので、１巡処理で出力単語ｇの値の状態が確定すれば、最少実行回数で処理が完了したと言える。

例５のシステムにおいて、ユーザが入力を行う前の単語の値の状態、すなわち単語の状態ベクトルＸの初期値は、Ｉ章で述べたように総ての要素が「null（未決定）」で、下記の数式２９のようになる。

【０１４８】

【数２９】

$$X=\begin{matrix}a\\b\\c\\d\\x_r\\e\\f\\g\end{matrix}\begin{bmatrix}\text{null}\\\text{null}\\\text{null}\\\text{null}\\\text{null}\\\text{null}\\\text{null}\\\text{null}\end{bmatrix}$$

数式２８の隣接行列Ｆ（すなわち例５のシステム）に、数式２９の状態ベクトルＸをかける計算処理を施す（すなわち、例５のシステムに入力を行う）。ＦＸの計算処理によって変化する単語の値の状態を１つの終点単語ごとに追って説明する。

（１）１巡目処理中の終点単語ａの計算

１巡目処理中の終点単語ａの計算は、次のようになる。

a

＝0・null＋0・null＋0・null ＋0・null＋0・null＋0・null＋0・null＋0・null

＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)

＝(+1)

入力単語aは始点単語を持たないので、どのような値の状態をかけても値の状態が確定し、nullから(+1)へ変化する。すなわち、これは入力によって値が確定したことを表わしている。その結果、１巡処理中の単語ａ行完了後の各単語の値の状態は、以下の表３のとおりである。

【０１４９】

【表３】

	a	b	c	d	x_r	e	f	g
値の状態	+1	null	null	null	null	null	null	null

（２）１巡目処理中の終点単語ｂの計算

１巡目処理中の終点単語ｂの計算は、表３の単語の値の状態を用いるので次のようになる。

b

＝0・(+1)＋0・null＋0・null ＋0・null＋0・null＋0・null＋0・null＋0・null

＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)

＝(+1)

入力単語ｂも始点単語を持たないので、どのような値の状態をかけても値の状態が確定し、nullから(+1)へ変化する。その結果、１巡処理中の単語ｂ行完了後の各単語の値の状態は、以下の表４のとおりである。

【０１５０】

【表４】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	null	null	null	null	null	null

例５のシステムでは、単語ａと単語ｂは画面上の２者択一の指示ボタンであるので、どちらか一方が選択されることによって、双方の値の状態が確定することになる。

（３）１巡目処理中の終点単語ｃの計算

１巡目処理中の終点単語ｃの計算は、表４の単語の値の状態を用いるので次のようになる。

b

＝0・(+1)＋0・(+1)＋0・null ＋0・null＋0・null＋0・null＋0・null＋0・null

＝(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1)

＝(+1)

入力単語ｃも始点単語を持たないので、どのような値の状態をかけても値の状態が確定し、nullから(+1)へ変化する。その結果、１巡処理中の単語ｃ行完了後の各単語の値の状態は、以下の表５のとおりである。

【０１５１】

【表５】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	+1	null	null	null	null	null

（４）１巡目処理中の終点単語ｄの計算

１巡目処理中の終点単語ｄの計算は、表５の単語の値の状態を用いるので次のようになる。

b

＝0・(+1)＋0・(+1)＋0・(+1) ＋0・null＋0・null＋0・null＋0・null＋0・null

＝(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1)

＝(+1)

入力単語ｄも始点単語を持たないので、どのような値の状態をかけても値の状態が確定し、nullから(+1)へ変化する。その結果、１巡処理中の単語ｄ行完了後の各単語の値の状態は、以下の表６のとおりである。

【０１５２】

【表６】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	+1	+1	null	null	null	null

（５）１巡目処理中の終点単語x_rの計算

１巡目処理中の終点単語x_rの計算は、表６の単語の値の状態を用いるので次のようになる。

x_r

＝1・(+1)＋1・(+1)＋1・(+1) ＋1・(+1)＋0・null＋0・null＋0・null＋0・null

＝(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1)

＝(+1)

単語ｘｒの始点単語となる単語ａ、ｂ、ｃ、ｄの値の状態がすでに確定した、すなわち、表２に示した経路作用要素である単語ｘｒの定義式実行条件（１）および（２）が満たされたので、単語ｘｒの値の状態もnullから確定済みの(+1)へ変化する。その結果、１巡処理中の単語ｘｒ行完了後の各単語の値の状態は、以下の表７のとおりである。

【０１５３】

【表７】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	+1	+1	+1	null	null	null

（６）１巡目処理中の終点単語ｅの計算

１巡目処理中の終点単語ｅの計算は、表７の単語の値の状態を用いるので次のようになる。

ｘｒ

＝0・(+1)＋0・(+1)＋1・(+1)　＋1・(+1)＋1・(+1)＋0・null＋0・null＋0・null

＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)

＝(+1)

単語ｅの始点単語となる単語ｃ、ｄおよびｘｒの値の状態がすでに確定済みであるので、すなわち、定義式ｃ＋ｄの始点単語の値が決定済み、かつ経路作用要素ｘｒが次に実行する基本構造を指定したので、単語ｅの値の状態もnullから確定済み(+1)へ変化する。その結果、１巡処理中の単語ｅ行完了後の各単語の値の状態は、以下の表８のとおりである。

【０１５４】

【表８】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	+1	+1	+1	+1	null	null

（７）１巡目処理中の終点単語ｆの計算

１巡目処理中の終点単語ｆの計算は、表８の単語の値の状態を用いるので次のようになる。

ｘｒ

＝0・(+1)＋0・(+1)＋1・(+1)　＋1・(+1)＋1・(+1)＋0・(+1)＋0・null＋0・null

＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)

＝(+1)

単語ｆの始点単語となる単語ｃ、ｄおよびｘｒの値の状態がすでに確定済みであるので、単語ｆの値の状態もnullから(+1)へ変化する。その結果、１巡処理中の単語ｆ行完了後の各単語の値の状態は、以下の表９のとおりである。

【０１５５】

【表９】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	+1	+1	+1	+1	+1	null

例５のシステムでは、単語ｅと単語ｆはそれぞれの実行条件が２者択一の条件であるので、どちらか一方の条件が真と確定すれば、他方の条件は偽と確定し、双方の値の状態が確定することになる。たとえば、ＢＳ２が指定されれば、単語ｅの定義式実行条件＝真が確定して単語ｅに値が与えられることによって値の状態が確定する。一方、単語ｆは、定義式実行条件＝偽が確定して単語ｆに値が与えられないことによって値の状態が確定する。従って、経路作用要素がどちらか一方の基本構造を指定したとき、単語ｅも単語ｆも値

の状態が確定済みとなるのである。

（８）１巡目処理中の終点単語ｇの計算

１巡目処理中の終点単語ｇの計算は、表９の単語の値の状態を用いるので次のようになる。

x_r

$=0\cdot(+1)+0\cdot(+1)+0\cdot(+1)+0\cdot(+1)+0\cdot(+1)+1\cdot(+1)+1\cdot(+1)+0\cdot\mathrm{null}$

$=(+1)+(+1)+(+1)+(+1)+(+1)+(+1)+(+1)+(+1)$

$=(+1)$

単語ｇの始点単語となる単語ｅおよびｆの値の状態がすでに確定済みであるので、単語ｇの値の状態もnullから(+1)へ変化する。その結果、１巡処理中の単語ｇ行完了後の各単語の値の状態は、以下の表１０のとおりである。

【０１５６】

【表１０】

	a	b	c	d	x_r	e	f	g
値の状態	+1	+1	+1	+1	+1	+1	+1	+1

以上で隣接行列Ｆの総ての単語行の計算処理が１巡したことになる。従って、表１０がＦＸの結果となる。下記の数式３０の状態ベクトルX_1がＦＸの結果である。

【０１５７】

【数３０】

$$X_1 = \begin{matrix} a \\ b \\ c \\ d \\ x_r \\ e \\ f \\ g \end{matrix} \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \end{bmatrix}$$

以上のように、隣接行列Ｆは、トポロジカルソートによって単語単位プログラムが最適順序で実行されるように並んでいるので、１巡の計算処理の実行で総ての単語の値の状態が確定し、出力単語ｇの値を成立することができた。

ＩＩ－２－（４）経路作用要素の排除

経路作用要素の実行条件は、経路作用要素によって指定される基本構造に属する全単語の集合（１つ以上からなる）の定義式を実行するための条件と等しい。なぜなら、その基本構造の単語の総ては、その経路作用要素の条件が満されたときに実行されるからである。従って、経路作用要素の実行条件は、経路作用要素によって指定される基本構造に属す単語の条件に移すことができる。

【０１５８】

例５のシステムの場合で具体的に述べると次のようになる。まず、経路作用要素である単語x_rと、単語ｅおよび単語ｆの定義式と定義式実行条件は下記の表１１の通りであった。

【０１５９】

【表１１】

単語	定義式	定義式実行条件
x_r	ＢＳ２を指定	（１）ａ＝真（ボタンａが押され、ａに値がある）かつ、ｃ＝真およびｄ＝真
	ＢＳ３を指定	（２）ｂ＝真（ボタンｂが押され、ｂに値がある）かつ、ｃ＝真およびｄ＝真
ｅ	ｃ＋ｄ	X_rがＢＳ２を指定
ｆ	ｄ×ｄ	X_rがＢＳ３を指定

従って、X_rの条件（１）「ａ＝真かつ、ｃ＝真およびｄ＝真」を、この条件が成立するとき指定する基本構造ＢＳ２に属する単語ｅの実行条件「ＸｒがＢＳ２を指定」と置き換え、条件（２）「ｂ＝真かつ、ｃ＝真およびｄ＝真」をこの条件が成立するとき指定する基本構造ＢＳ３に属する単語ｆの実行条件「ＸｒがＢＳ３を指定」と置き換える、ということになる。この結果、単語ｅおよび単語ｆの定義式、定義式実行条件は下記の表１２のようになる。

【０１６０】

【表１２】

単語	定義式	定義式実行条件
ｅ	ｃ＋ｄ	ａ＝真（ボタンａが押され、ａに値がある）かつ、ｃ＝真およびｄ＝真
ｆ	ｃ×ｄ	ｂ＝真（ボタンｂが押され、ｂに値がある）かつ、ｃ＝真およびｄ＝真

表１２のように、経路作用要素の実行条件を、指定先の基本構造の単語の実行条件に置き換えると、ＢＳ１とＢＳ２およびＢＳ３のリンクは、ＢＳ２およびＢＳ３の単語の実行条件によって成立するので、経路作用要素は単に、次の基本構造の実行に移る条件が整ったこと（入力単語ｃおよびｄに値があり、かつボタンａまたはｂが押された）を示すだけの意味をしか持たなくなる。これは、経路作用要素を取り除いてもシステムの意味に変化が生じないことを意味する。従って、経路作用要素の実行条件を、その条件の成立によって指定される基本構造の単語の実行条件に置き換えを行った後、システム全体の隣接行列から、経路作用要素を除くことができる。

【０１６１】

図１４は、経路作用要素を取り除いて、例５のシステムの単語の関係を有向グラフで表したものである。図１４では、経路作用要素である単語x_rがなくなり、単語eおよび単語ｆと始点単語との関係は、上記の表１２で定義された定義式と定義式実行条件に従って示されている。すなわち、単語ｅは、始点単語として単語a（定義式実行条件「実行ボタンａが押された」）、単語ｃおよび単語ｄ（定義式「ｃ＋ｄ」および、定義式実行条件「ｃ＝真およびｄ＝真」）を持つ。単語ｆは、始点単語として単語ｂ（定義式実行条件「実行ボタンｂが押された」）、単語ｃおよび単語ｄ（定義式「ｃ＋ｄ」および、定義式実行条件「ｃ＝真およびｄ＝真」）を持つ。単語ｅおよび単語ｆから単語ｇへの関係は図１１と変らない。

【０１６２】

経路作用要素を取り除いた例５のシステムを、図１４の有向グラフに従って隣接行列に表わせば、以下のようになる。数式３１のＦ１’は、基本構造ＢＳ１（入力）の隣接行列である。経路作用要素である単語x_rが要素から削除されている。

【０１６３】

【数３１】

$$F1' = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

数式３２のＦ２’は、基本構造ＢＳ２＆ＢＳ３の隣接行列である。

【０１６４】

【数３２】

$$F2' = \begin{array}{c} \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cc} e & f \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{array}$$

数式３３のＦ３’は、基本構造ＢＳ１（出力）の隣接行列である。

【０１６５】

【数３３】

$$F3' = \begin{array}{c} g \\ g \end{array} \begin{bmatrix} 0 \end{bmatrix}$$

数式３４のＦＣ１’は、基本構造ＢＳ１（入力）から基本構造ＢＳ２＆ＢＳ３への関係を示す結合隣接行列である。経路作用要素が削除されたので、始点単語の要素として単語X_rが削除されている。単語ｅおよび単語ｆは、表１２に示したように経路作用要素の実行条件を移したことで、それぞれ新たに単語ａ、単語ｂを始点単語として持つことになった。従って、終点単語ｅは、始点単語ｃおよびｄの他に単語ａとの交点が１となっている。終点単語ｆは、始点単語ｃおよびｄの他に単語ｂとの交点が１となっている。

【０１６６】

【数３４】

$$FC1' = \begin{array}{c} \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

数式３５のＦＣ２’は、基本構造ＢＳ２＆ＢＳ３から基本構造ＢＳ１（出力）への関係を示す結合隣接行列である。経路作用要素が削除されたので、始点単語の要素として単語X_rが削除されている。

【０１６７】

【数３５】

$$FC2' = g\begin{bmatrix} a & b & c & d & e & f \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

数式３６のＦＣ３’は、基本構造ＢＳ２＆ＢＳ３から基本構造ＢＳ１（入力）への関係がないことを示す結合隣接行列である。経路作用要素が削除されたので、始点単語の要素として単語X_rが削除されている。

【０１６８】

【数３６】

$$FC3' = \begin{matrix} a \\ b \\ c \\ d \end{matrix}\begin{bmatrix} e & f \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

数式３７のＦＣ４’は、基本構造ＢＳ１（出力）から基本構造ＢＳ１（入力）への関係がないことを示す結合隣接行列である。経路作用要素が削除されたので、始点単語の要素として単語X_rが削除されている。

【０１６９】

【数３７】

$$FC4' = \begin{matrix} a \\ b \\ c \\ d \end{matrix}\begin{bmatrix} g \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

数式３８のＦＣ５’は、基本構造ＢＳ１（出力）から基本構造ＢＳ２＆ＢＳ３への関係がないことを示す結合隣接行列である。

【０１７０】

【数３８】

$$FC5' = \begin{matrix} e \\ f \end{matrix}\begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix}$$

システムの隣接行列Ｆは、経路作用要素の条件の置き換えと経路作用要素の削除を行うと、下記の数式３９に示す隣接行列Ｆ’で表わすことができる。

【０１７１】

【数３９】

$$F'=\begin{bmatrix} F1' & 0 & 0 \\ FC1' & F2' & 0 \\ FC2' & & F3' \end{bmatrix}=\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array}\begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

上記数式で、「Ｆ’＝」の右辺に示された、Ｆ１’、Ｆ２’、Ｆ３’、ＦＣ１’、ＦＣ２’と０（行列の交点が総て０である隣接行列で、ＦＣ３’、ＦＣ４’、ＦＣ５’に相当する）で構成された行列は、基本構造単位の隣接行列と結合隣接行列がどのように統合されてシステム全体を表わすかを示している。次に、それらの要素を単語に変えたものが、その右辺にあるａ、ｂ、ｃ、ｄ、ｅ、ｆ、ｇを行列の要素とする隣接行列である。Ｆ１’、Ｆ２’、Ｆ３’、ＦＣ１’、ＦＣ２’、ＦＣ３’、ＦＣ４’、ＦＣ５’がどのように統合されてＦ’を構成するのか、さらにわかりやすく示したのが図１５である。図１５の１２０１の部分がＦ１’、１２０２がＦ２’、１２０３がＦ３’、１２０４がＦＣ１’、１２０５がＦＣ２’である。行列の交点が総て０である結合隣接行列は、ＦＣ３’が１２０６、ＦＣ４’が１２０７、ＦＣ５’が１２０８を構成する。

【０１７２】

以上のようにして、２つ以上の基本構造から成る１つのシステムを、経路作用要素を排除して、１つの隣接行列で表わすことができる。

次に、このシステムの隣接行列に対して、トポロジカルソートを行って、総ての単語の値の状態が最少実行回数で確定（すなわち総ての出力単語の生成が完了）するような最適順序に単語単位のプログラムの並び替えを行う。隣接行列Ｆ’をみると、図１６に示すように、値が１である始点単語との交点は総て左下三角形（１６０１）内にある。Ｉ章で述べたように、これは単語が最適順序に並んでいることを意味する。従って、数式３９の隣接行列Ｆ’は、すでにトポロジカルソート済みであって、初期値の単語の状態ベクトルを与えれば、最少実行回数で出力単語の生成を完了することができる単語の順序になっているといえる。

ＩＩ－２－（５）　経路作用要素を排除した構造の検証

このセクションでは、前セクションで示した、経路作用要素を排除したシステムの構造を示す隣接行列Ｆ’が、経路作用要素を排除する前の構造である隣接行列Ｆと同じ関数機能を持つことを検証する。以下に、隣接行列Ｆ’に単語の状態ベクトルを与えて出力単語ｇの値を求める過程を示す。ｍ回目の計算処理の結果、単語の状態ベクトルX_mの値が総て確定済みとなれは、Ｆ’もＦと同等に機能する関数だと言える。また、トポロジカルソートを施したことによって、最少実行回数で出力単語の生成を完了できることも検証する。例５のシステムであるＦ’も、1巡処理で単語ｇの値の状態が確定すれば、最少実行回数で実行できたと言える。

前述したように、単語の状態ベクトルの初期値は総ての要素が「null（未決定）」であるので、Ｆ’に最初に与えるべき単語の状態ベクトルＸ’は下記の数式４０のようになる。

【０１７３】
【数４０】

$$X' = \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix} \begin{bmatrix} \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \end{bmatrix}$$

数式３９の隣接行列Ｆ’（すなわち例５のシステム）に、数式３９の状態ベクトルＸ’をかける計算処理を施す（すなわち、例５のシステムに入力を行う）。Ｆ’Ｘ’の計算処理によって変化する単語の値の状態を１つの終点単語ごとに追って説明する。

（１）Ｆ’１巡目処理中の終点単語ａの計算

終点単語ａは、計算処理にＸ’の単語の値の状態を用いるので、１巡目処理中の終点単語ａの計算は、次のようになる。

a
＝0・null＋0・null＋0・null ＋0・null＋0・null＋0・null＋0・null
＝(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1)
＝(+1)

入力単語aは始点単語を持たないので、値の状態が確定し、nullから(+1)へ変化する。すなわち、これは入力によって値が確定したことを表わしている。その結果、１巡処理中の単語ａ行完了後の各単語の値の状態は、以下の表１３のとおりである。

【０１７４】
【表１３】

	a	b	c	d	e	f	g
値の状態	+1	null	null	null	null	null	null

（２）Ｆ’１巡目処理中の終点単語ｂの計算

終点単語ｂは、計算処理に表１３の単語の値の状態を用いるので、１巡目処理中の終点単語ｂの計算は、次のようになる。

b
＝0・(+1)＋0・null＋0・null ＋0・null＋0・null＋0・null＋0・null
＝(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1) ＋(+1)
＝(+1)

入力単語ｂも始点単語を持たないので、値の状態が確定し、nullから(+1)へ変化する。その結果、１巡処理中の単語ａ行完了後の各単語の値の状態は、以下の表１４のとおりである。

【０１７５】
【表１４】

	a	b	c	d	e	f	g
値の状態	+1	+1	null	null	null	null	null

（３）Ｆ’１巡目処理中の終点単語ｃの計算

終点単語ｃは、計算処理に表１４の単語の値の状態を用いるので、１巡目処理中の終点単語ｃの計算は、次のようになる。

c

＝0・(+1)＋0・(+1)＋0・null ＋0・null＋0・null＋0・null＋0・null
＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)
＝(+1)

入力単語ｃも始点単語を持たないので、値の状態が確定し、nullから(+1)へ変化する。その結果、１巡処理中の単語ｃ行完了後の各単語の値の状態は、以下の表１５のとおりである。

【０１７６】
【表１５】

	a	b	c	d	e	f	g
値の状態	+1	+1	+1	null	null	null	null

（４）Ｆ’１巡目処理中の終点単語ｄの計算

終点単語ｄは、計算処理に表１５の単語の値の状態を用いるので、１巡目処理中の終点単語ｄの計算は、次のようになる。

d
＝0・(+1)＋0・(+1)＋0・(+1) ＋0・null＋0・null＋0・null＋0・null
＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)
＝(+1)

入力単語ｄも始点単語を持たないので、値の状態が確定し、nullから(+1)へ変化する。その結果、１巡処理中の単語ｄ行完了後の各単語の値の状態は、以下の表１６のとおりである。

【０１７７】
【表１６】

	a	b	c	d	e	f	g
値の状態	+1	+1	+1	+1	null	null	null

（５）Ｆ’１巡目処理中の終点単語ｅの計算

終点単語ｅは、計算処理に表１６の単語の値の状態を用いるので、１巡目処理中の終点単語ｅの計算は、次のようになる。

e
＝1・(+1)＋0・(+1)＋1・(+1) ＋1・(+1)＋0・null＋0・null＋0・null
＝(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)　＋(+1)
＝(+1)

単語ｅの始点単語である単語ａ、ｃ、ｄの値の状態が確定しているので、単語ｅの値の状態もnullから(+1)へ変化して確定する。その結果、１巡処理中の単語ｅ行完了後の各単語の値の状態は、以下の表１７のとおりである。

【０１７８】
【表１７】

	a	b	c	d	e	f	g
値の状態	+1	+1	+1	+1	+1	null	null

（６）Ｆ’１巡目処理中の終点単語ｆの計算

終点単語ｆは、計算処理に表１７の単語の値の状態を用いるので、１巡目処理中の終点単語ｆの計算は、次のようになる。

f

＝0・(+1)＋1・(+1)＋1・(+1)　＋1・(+1)＋0・(+1)＋0・null＋0・null
　＝(+1)　＋(+1)　＋(+1)　　＋(+1)　＋(+1)　＋(+1)　＋(+1)
　＝(+1)

単語ｆの始点単語である単語ｂ、ｃ、ｄの値の状態が確定しているので、単語ｆの値の状態もnullから(+1)へ変化して確定する。その結果、１巡処理中の単語ｆ行完了後の各単語の値の状態は、以下の表１８のとおりである。

【０１７９】

【表１８】

	a	b	c	d	e	f	g
値の状態	+1	+1	+1	+1	+1	+1	null

（７）Ｆ’１巡目処理中の終点単語ｇの計算

終点単語ｇは、計算処理に表１８の単語の値の状態を用いるので、１巡目処理中の終点単語ｇの計算は、次のようになる。

g
＝0・(+1)＋0・(+1)＋0・(+1)　＋0・(+1)＋1・(+1)＋1・(+1)＋0・null
　＝(+1)　＋(+1)　＋(+1)　　＋(+1)　＋(+1)　＋(+1)　＋(+1)
　＝(+1)

単語ｇの始点単語である単語ｅ、ｆの値の状態が確定しているので、単語ｇの値の状態もnullから(+1)へ変化して確定する。その結果、１巡処理中の単語ｇ行完了後の各単語の値の状態は、以下の表１９のとおりである。

【０１８０】

【表１９】

	a	b	c	d	e	f	g
値の状態	+1	+1	+1	+1	+1	+1	+1

以上で隣接行列Ｆ’の総ての単語行の計算処理が１巡したことになる。従って、表１９の単語の値の状態がＦ’Ｘ’の結果となる。下記の数式４１の状態ベクトルＸ’$_1$がＦ’Ｘ’の結果である。

【０１８１】

【数４１】

$$X'_1 = \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix} \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \end{bmatrix}$$

以上のように、隣接行列Ｆ’は、トポロジカルソートによって単語単位プログラムが最適順序で実行されるように並んでいるので、１巡の計算処理の実行で総ての単語値が確定し、出力単語ｇの値を成立することができた。

また、経路作用要素の単語を除いた隣接行列Ｆ’も、状態ベクトルの初期値を与える（入力を行う）ことで、隣接行列Ｆと同様に総ての単語の値の状態が確定し、出力単語ｇの値が生成できたことがわかる。従って、経路作用要素のあるシステムである隣接行列Ｆと、経路作用要素のないシステムである隣接行列Ｆ’は、システムとして同等の処理機能を

持つことが検証できた。

上記で述べたように、全体システムが複数の基本構造を有する場合に、これを統合し、さらに当該統合されたものに対して（例えばトポロジカルソートのような）探索を施すことで、Ｌｙｅｅ（登録商標）等のソフトウェアの自律的開発技術の適用に際して各段のパフォーマンスを提供する。これが例えば、統合されずに複数の基本構造をそのまま有する場合には、本願に係る発明によるほどのパフォーマンスは得られず、また、トポロジカルソート等の探索技法によって整序しなかった場合には本願に係る発明によるほどのパフォーマンスは得られない。

ＩＩＩ章　むすび

処理経路図を作成すること、すなわち分割してとらえることは、システムの要件を正確にとらえるために非常に有用である。処理経路図を作成することは、出力条件に共通要素を持つ単語を論理レコードの単位でとらえることである。条件の共通要素を経路作用要素に割り当てることは、効率的にユーザ要件を定義するためには大きなメリットがある。

一方、プログラムコードを生成し、実行する時点では、基本構造に分割したシステムを統合することが有用である。なぜなら、システム全体の単語単位のプログラムの実行順序を、最少実行回数で出力単語の生成を完了できる順序に並べることができるからである。

本明細書では、単語単位のプログラムからなるシステムにおける、出力データを生成するための単語単位プログラムの処理を、無駄な繰り返しを排除して最少実行回数で完了することを課題として、以下の手段で解決することについて述べた。

１）同じ出力条件を持つ単語の集合（論理レコード）を、１つの基本構造として、システム全体を処理経路図でとらえる。これによって、同じ基本構造に属する単語単位のプログラムに共通する実行条件（経路作用要素がその基本構造を指定する条件）を的確にとらえることができる。

２）経路作用要素も単語の１つみなし、単語の関係を定義している単語単位のプログラムとして、要件を定義する。すなわち、どのような条件が成立したとき（定義式実行条件）、どの基本構造の実行を指定するか（定義式）、を定義する。

３）この経路作用要素の定義式実行条件を、その条件が成立するときに経路作用要素が指定する基本構造に属する単語の定義式実行条件に置くことによって、経路作用要素を排除する。経路作用要素の実行条件を、指定先の基本構造の単語の実行条件に置くことによって、経路作用要素を削除しても、基本構造間のリンク（すなわち単語間のリンク）が成立するので、システム全体を１つにまとめることが可能になる。

４）こうして得られた単一化されたシステム全体の単語単位のプログラム群（経路作用要素を含めない）に対して、トポロジカル・ソートを行って単語単位のプログラム群を最適順序に並び替える。これにより、無駄な繰返しを回避し、最少実行回数でプログラムを実行することが可能となる。

本発明は当該技術分野における通常の知識を有する者にとって修正および改変が容易に数多くなし得るので、図示および記述されたものと寸分違わぬ構成および動作に本発明を限定することは望ましくないことであり、従って、あらゆる適切な改変体および等価体は本発明の範囲に含まれるものと見なされうる。前述の本発明に係る実施の具体的形態の説明および例示によって詳細に記述されたが、本願の特許請求の範囲のみならず本発明に係る開示事項全体に定義された本発明の範囲から逸脱することなしに、修正、置換、および

、変更が数多く可能である。

【０１８２】

また、本願に係る発明は、その適用において、上記の記述において説明されるか、或いは、図面に示された要素の詳細な解釈及び組み合わせに限定されるものではない。本発明は、他の実施形態が可能であり、種々の方法で実用および実施可能である。また、ここで用いられた語法および用語は記述を目的とするものであり、限定的に働くものとみなされてはならない。

【０１８３】

従って、当該技術分野における通常の知識を有する者は、本開示の基調となる概念は、本発明の幾つかの目的を実施するための他の構造、方法、及び、システムを設計するための基礎として容易に利用され得ることを理解するはずである。従って、本発明の趣旨および範囲から逸脱しない限り、本願の特許請求の範囲にはそのような等価な解釈が含まれるものと見なされるものである。

【０１８４】

上記の詳細な説明では、有向グラフ表記、隣接行列算出、トポロジカル・ソート、並び替えという順序で本発明の思想を実現する方法について述べた。しかしこれらはどれも絶対不可欠な要素というものではなく、例えば、有向グラフ表記の表記を経ないで直接隣接行列算出→トポロジカル・ソート→並び替えという順序で本発明の思想を実現してもよい。

【０１８５】

さらに、有向グラフに対して、例えば幅優先探索（Ｂｒｅａｄｔｈ－Ｆｉｒｓｔ　Ｓｅａｒｃｈ）、反復深化法（Ｉｔｅｒａｔｉｖｅ　Ｄｅｅｐｉｎｇ）、発見的探索（Ｈｅｕｒｉｓｔｉｃ　Ｓｅａｒｃｈ）、ヒル・クライミング（Ｈｉｌｌ　Ｃｌｉｍｂｉｎｇ）、最適優先探索（Ｂｅｓｔ－Ｆｉｒｓｔ　Ｓｅａｒｃｈ）、オイラー（Ｅｕｌｅｒ）の一筆書き、ダイクストラ（Ｄｉｊｋｓｔｒａ）法等の各種探索技法を組み合わせることで並び替えてもよい。

【０１８６】

或いは、隣接行列算出のあとを、これらの各種探索技法を組み合わせることで本発明の根本思想を実現してもよい。

【０１８７】

また、本発明に係る技術思想は、例えばコンピュータソフトウェアの自動開発装置、自動開発プログラム、自動開発プログラムを記録した記録媒体、伝送媒体、紙媒体としても、また、自動開発プログラムを登載したコンピュータ・装置、自動開発プログラムを実行するクライアント・サーバ形式等といったカテゴリーにおいても実現、利用可能であることはいうまでもない。

【０１８８】

さらに本発明は、単一プロセッサ、単一ハードディスクドライブ、及び、単一ローカルメモリを備えたコンピュータシステムに限らず、当該システムのオプションとして、任意の複数または組み合わせプロセッサ又は記憶デバイスを装備するにも適している。コンピュータシステムは、精巧な計算器、掌タイプコンピュータ、ラップトップ／ノートブックコンピュータ、ミニコンピュータ、メインフレームコンピュータ、及び、スーパーコンピュータ、ならびに、これらの処理システムネットワーク組合わせを含む。本発明の原理に従って作動する任意の適切な処理システムによって代替されうるし、また、これらと組合せて用いることも可能である。

【０１８９】

また、本発明に係る技術思想は、もとよりあらゆる種類のプログラミング言語に対応可能である。さらに本発明に係る技術思想は、あらゆる種類・機能のアプリケーションソフトウェアに対しても適用可能である。

【０１９０】

またさらに本願発明は、その技術思想の同一及び等価に及ぶ範囲において様々な変形、

追加、置換、拡大、縮小等を許容するものである。また、本願発明を用いて生産されるソフトウェアが、その２次的生産品に登載されて商品化された場合であっても、本願発明の価値は何ら減ずるものではない。

【産業上の利用可能性】

【０１９１】

上記で規定される本発明では、繰返しをなくす前処理を自動で行い、こうして得られた前処理済の単語単位のプログラムをもとにＬｙｅｅ（登録商標）方法論により自動的に目的プログラム生成を行う。つまり、ユーザ要件の洗練化（整列）から目的プログラムの生成に至るまで自動化することが可能となる。これにより、ソフトウェア生産の大幅な効率向上、生産性向上、品質向上等、ソフトウェア産業上に大きな効果をもたらす。

【図面の簡単な説明】

【０１９２】

【図１】状態単語αについての数式３を計算するプログラムの構造を示したフローチャートである。

【図２】ａ≦ｂ(aからｂが作られる)という関係をもつ順序を説明するための有向グラフの例である。

【図３】有向グラフの例である。

【図４】あるノード１乃至８が形成する特定の半順序関係を表した有向グラフである。

【図５】トポロジカル・ソートの動作を説明するためのフローチャートである。

【図６】Ｌｙｅｅ（登録商標）方法論における単語単位の論理要素をノードとした場合にトポロジカル・ソートを用いて繰り返しをなくすための自動制御の動作を説明するためのフローチャートである。

【図７】本発明をソフトウェア開発装置として実現する場合の機能構成を示したブロック図である。

【図８】処理経路図の例である。

【図９】例５のシステムの画面の説明図である

【図１０】例５のシステムの単語を基本構図単位に示した説明図である。

【図１１】例５のシステムの単語の関係を表わした有向グラフである。

【図１２】例５のシステム全体を１つの隣接行列Ｆで表わす過程を説明するための図である。

【図１３】隣接行列Ｆがトポロジカルソート済みであることを説明するための図である。

【図１４】例５のシステムの構造から経路作用要素を削除したときの、単語の関係を表わした有向グラフである。

【図１５】例５のシステムの構造から経路作用要素を削除したとき、システム全体を１つの隣接行列Ｆ’で表わす過程を説明するための図である。

【図１６】隣接行列Ｆ’がトポロジカルソート済みであることを説明するための図である。

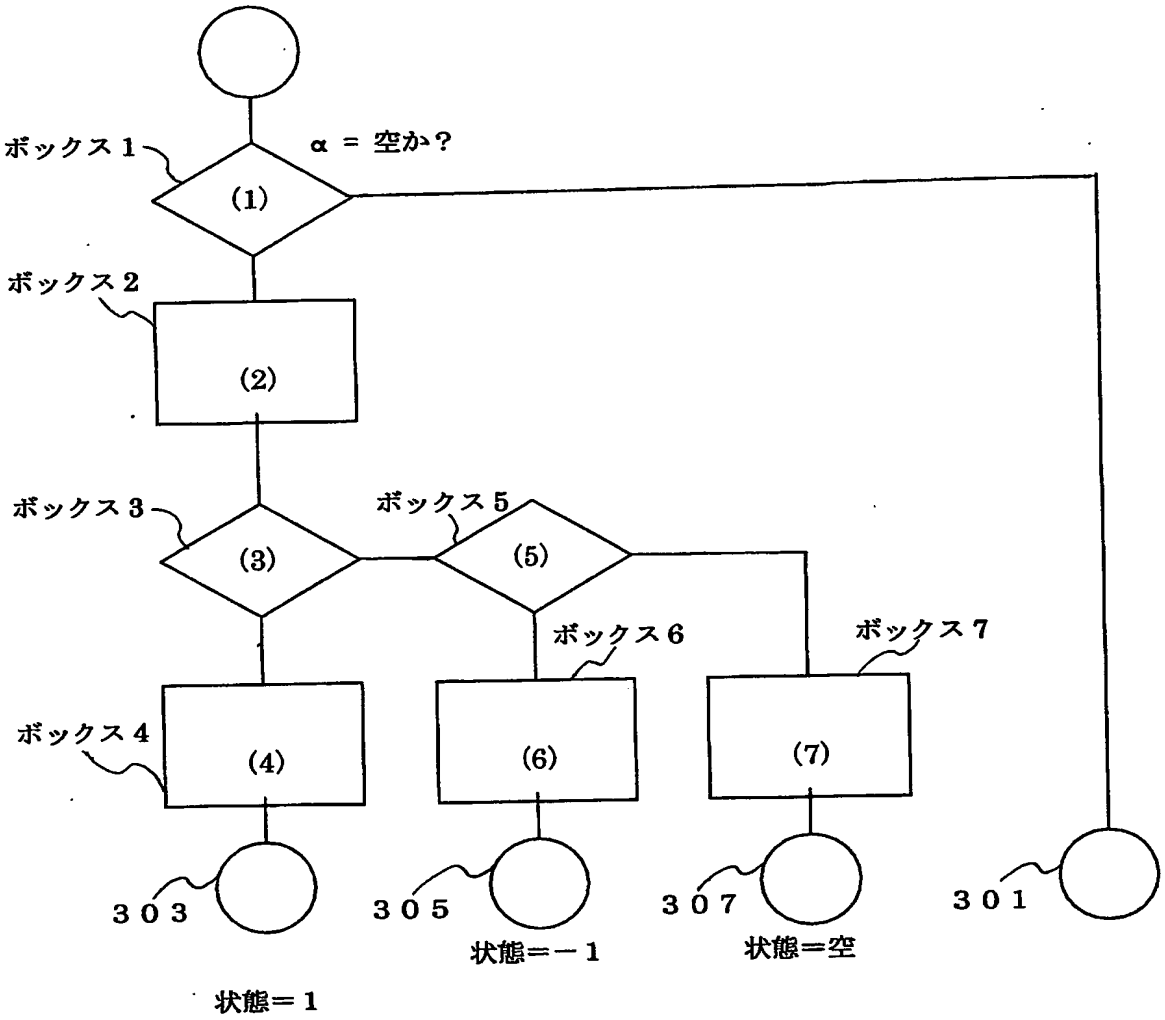
【符号の説明】

【０１９３】

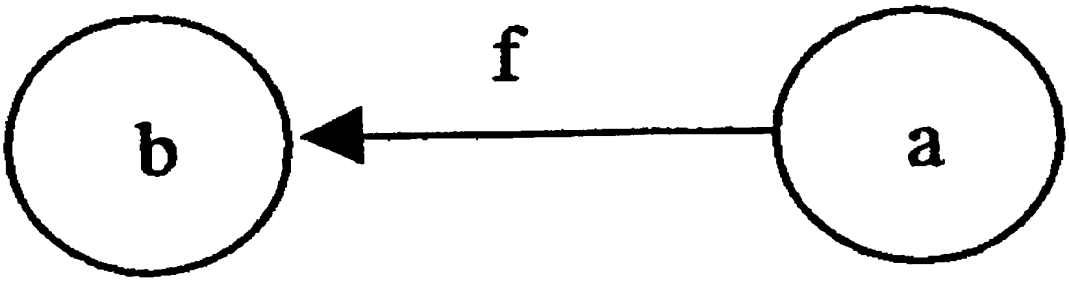
１１０１　基本構造ＢＳ１（入力）
１１０２　基本構造ＢＳ２
１１０３　基本構造ＢＳ３
１１０４　基本構造ＢＳ１（出力）
１２０１　隣接行列Ｆ１’
１２０２　隣接行列Ｆ２’
１２０３　隣接行列Ｆ３’
１２０４　結合隣接行列ＦＣ１’

１２０５　結合隣接行列ＦＣ２’

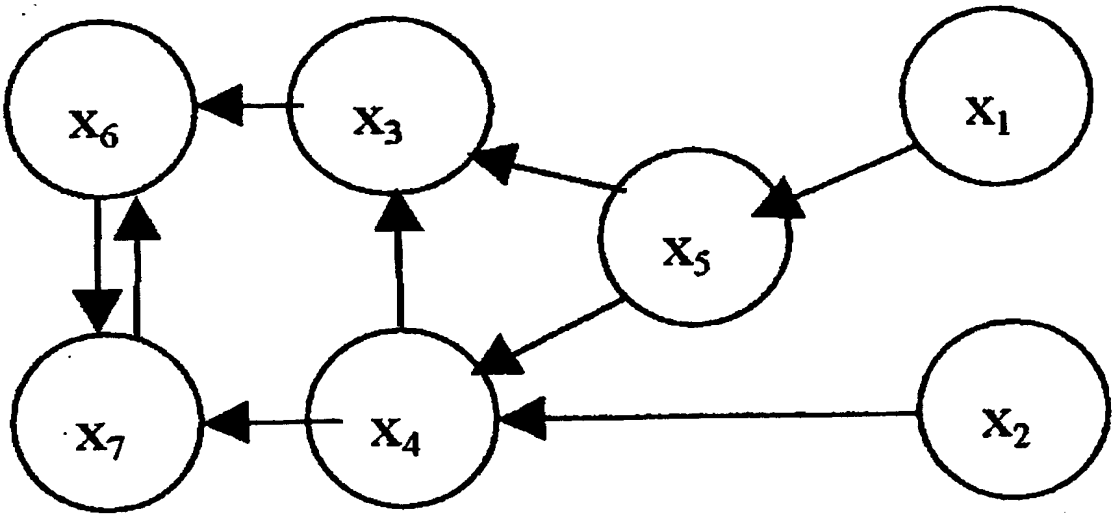
【書類名】図面
【図１】


ボックス１
α = 空か？
(1)
ボックス２
(2)
ボックス３
(3)
ボックス５
(5)
ボックス６
ボックス７
ボックス４
(4)
(6)
(7)
３０３
３０５
３０７
３０１
状態＝－１
状態＝空
状態＝１

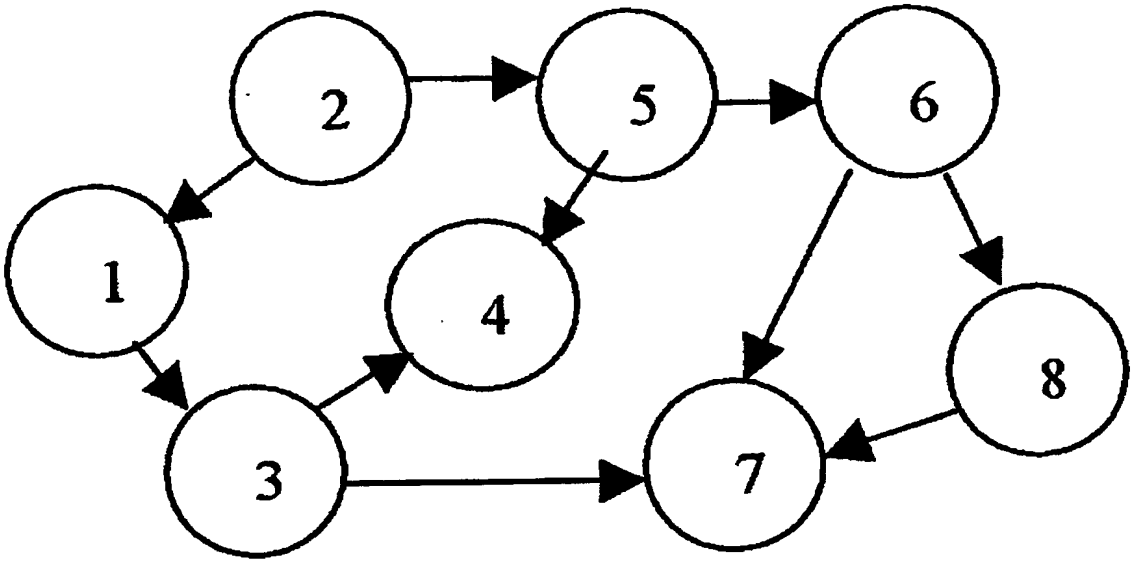
【図２】


f
b
a

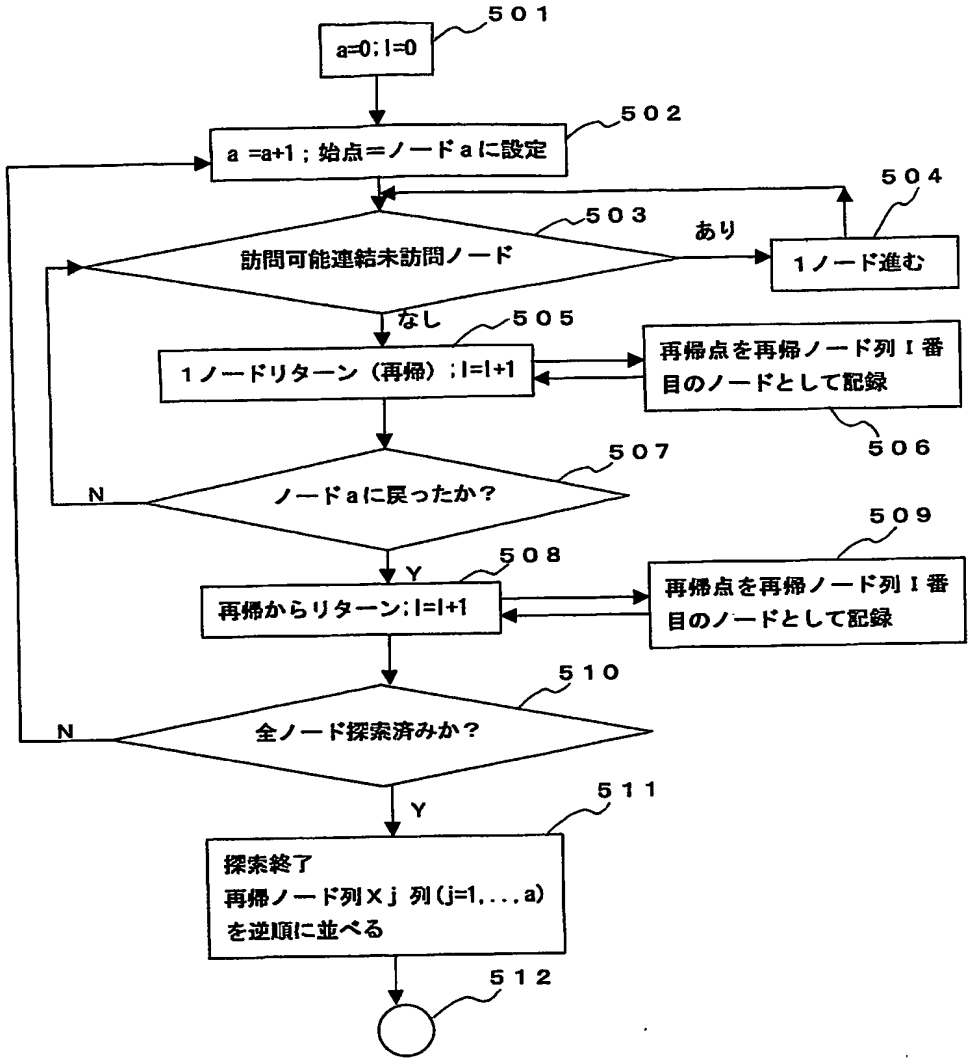
【図３】



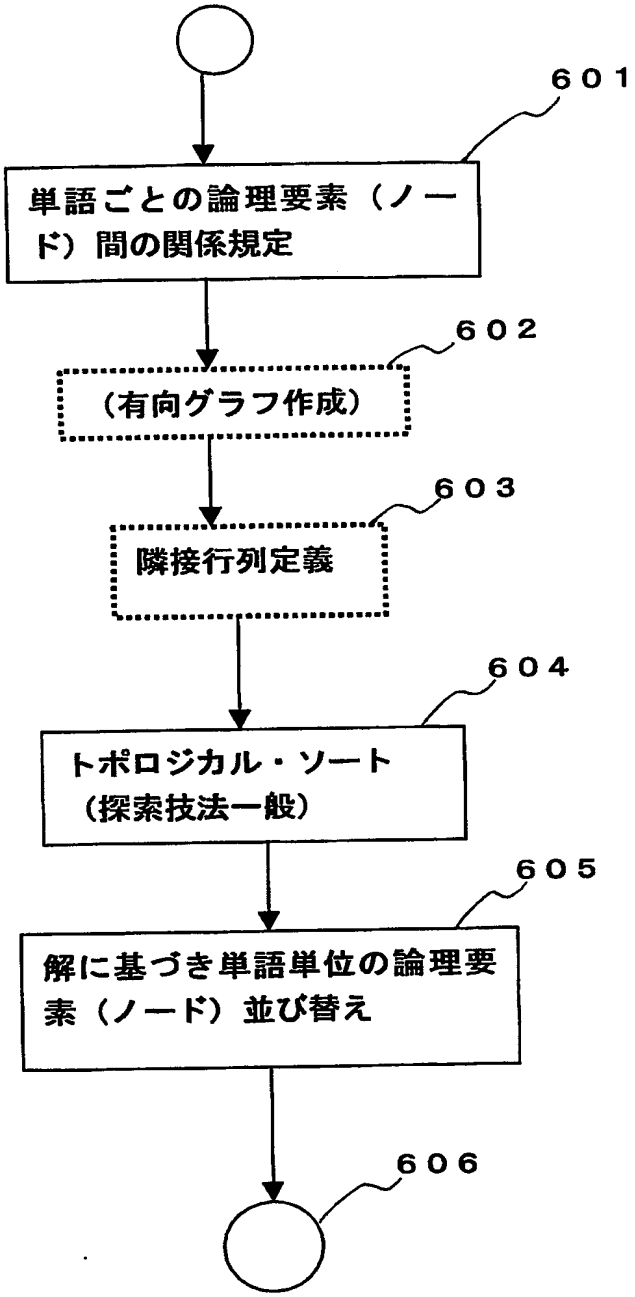
【図４】



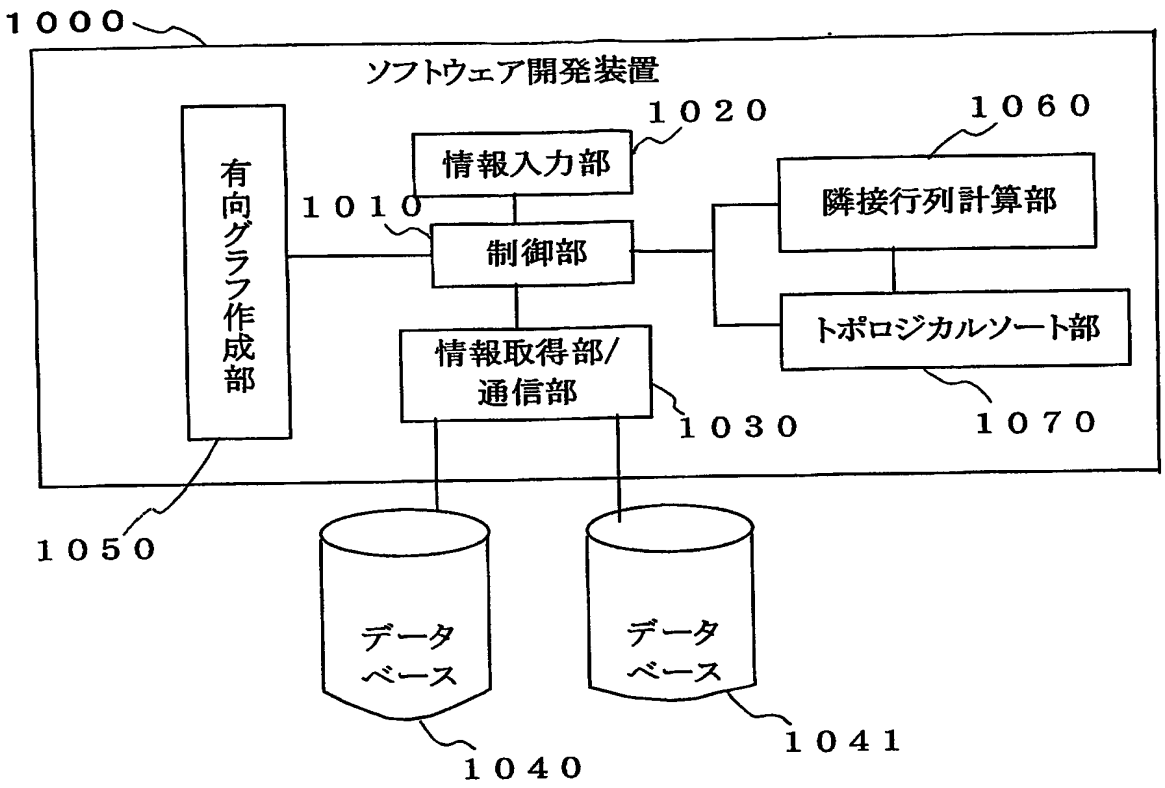
【図５】


501
a=0;l=0
502
a =a+1；始点＝ノードａに設定
504
503
あり
訪問可能連結未訪問ノード
1ノード進む
なし
505
1ノードリターン（再帰）；l=l+1
再帰点を再帰ノード列Ｉ番目のノードとして記録
506
507
N
ノードａに戻ったか？
509
508
Y
再帰からリターン；l=l+1
再帰点を再帰ノード列Ｉ番目のノードとして記録
510
N
全ノード探索済みか？
511
Y
探索終了
再帰ノード列Ｘｊ 列（j=1,...a）
を逆順に並べる
512

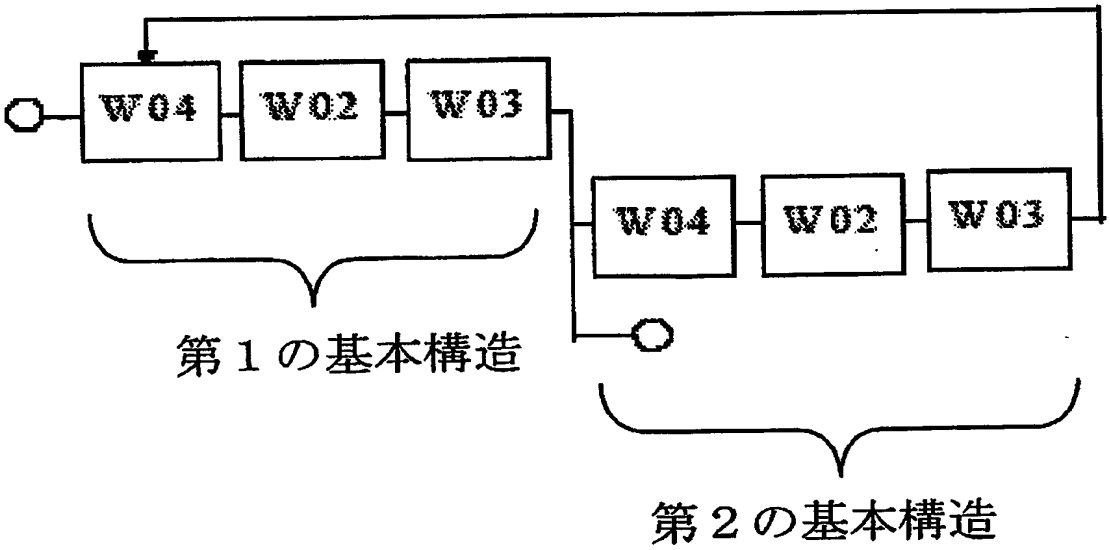
【図６】


601
単語ごとの論理要素（ノード）間の関係規定
602
（有向グラフ作成）
603
隣接行列定義
604
トポロジカル・ソート
（探索技法一般）
605
解に基づき単語単位の論理要素（ノード）並び替え
606

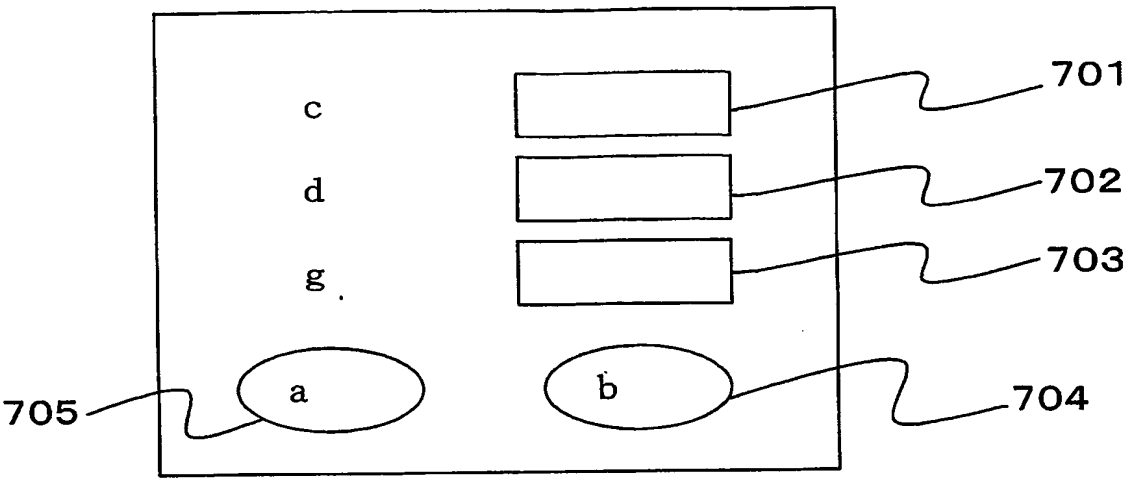
【図７】


１０００
ソフトウェア開発装置
有向グラフ作成部
１０２０
情報入力部
１０１０
制御部
１０６０
隣接行列計算部
トポロジカルソート部
情報取得部/
通信部
１０３０
１０７０
１０５０
データ
ベース
データ
ベース
１０４０
１０４１

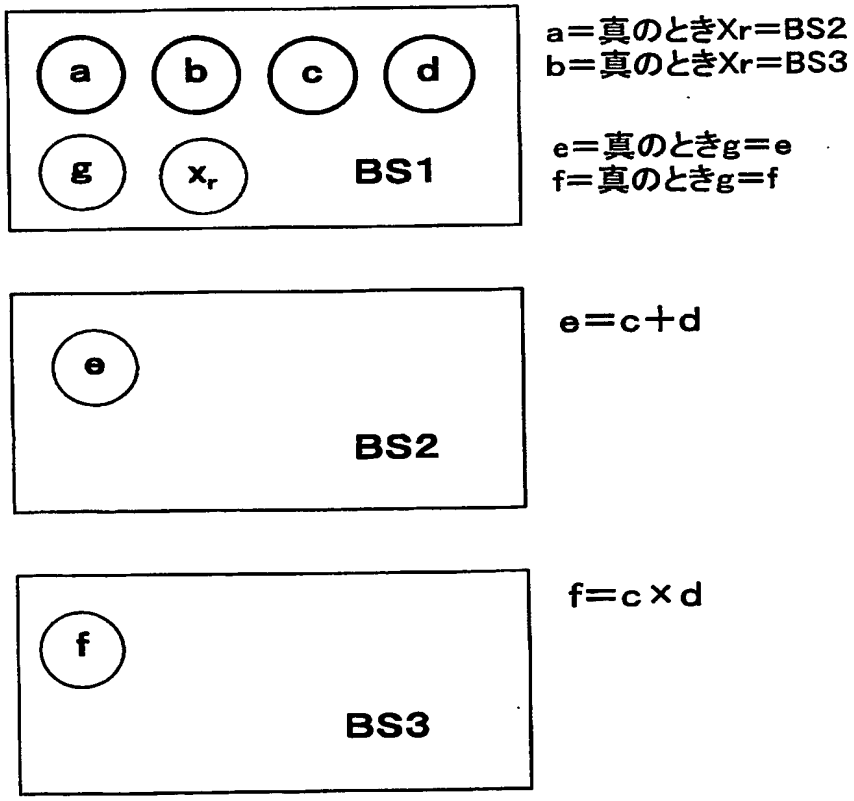
【図８】


W04
W02
W03
W04
W02
W03
第１の基本構造
第２の基本構造

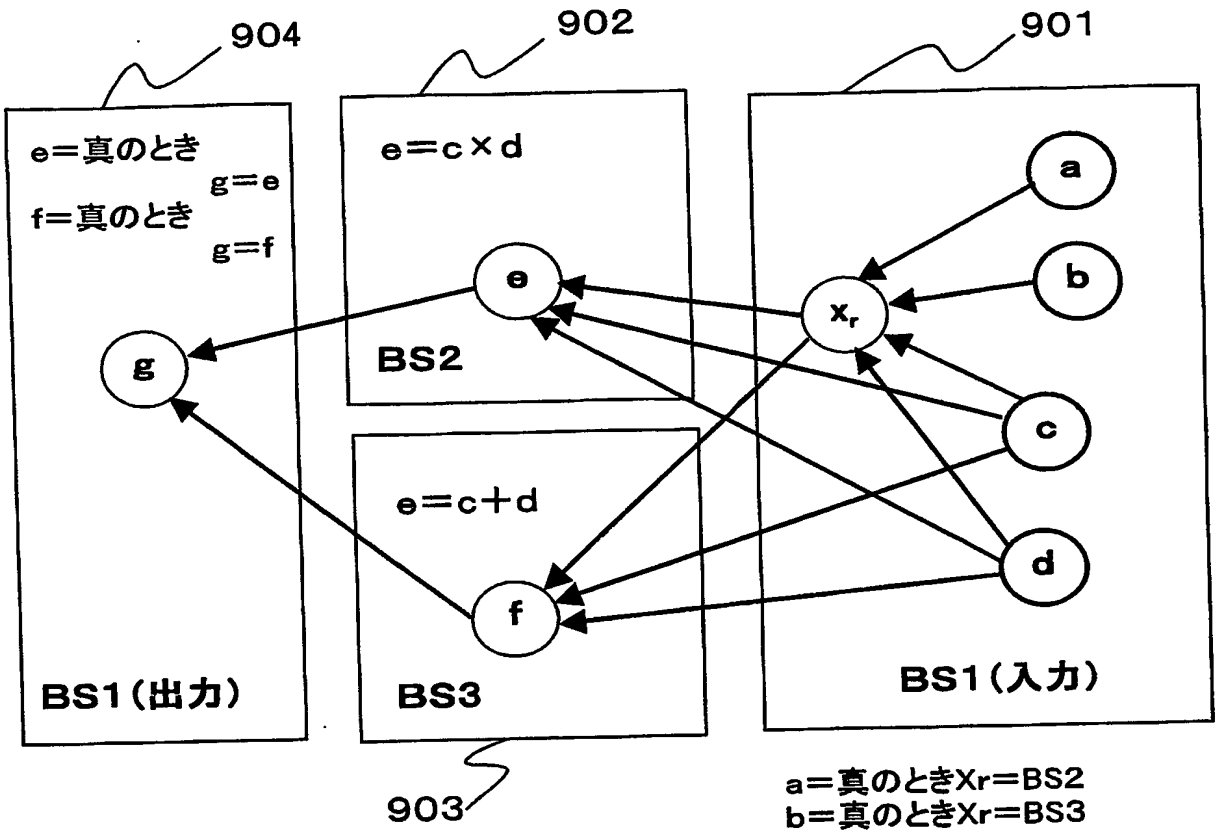
【図９】


c
d
g
a
b
701
702
703
704
705

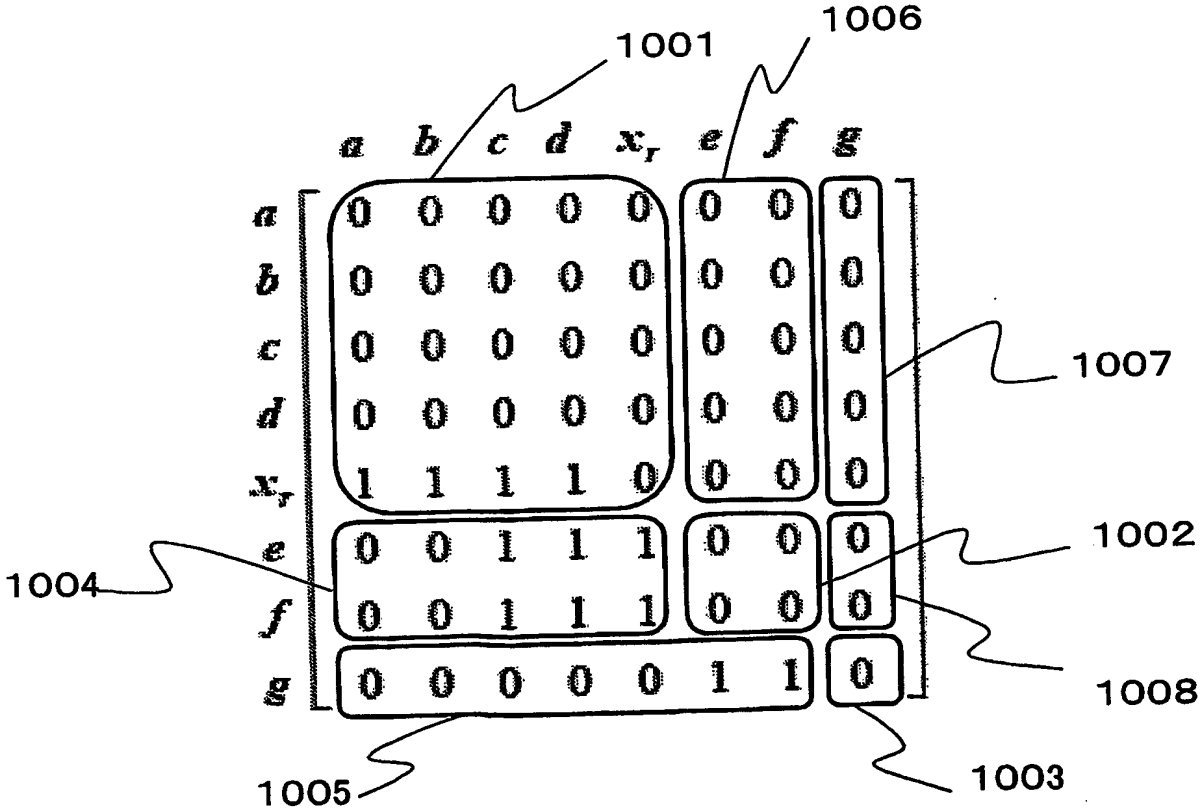
【図１０】


a
b
c
d
g
x_r
BS1
a＝真のときXr＝BS2
b＝真のときXr＝BS3
e＝真のときg＝e
f＝真のときg＝f
e
BS2
e＝c＋d
f
BS3
f＝c×d

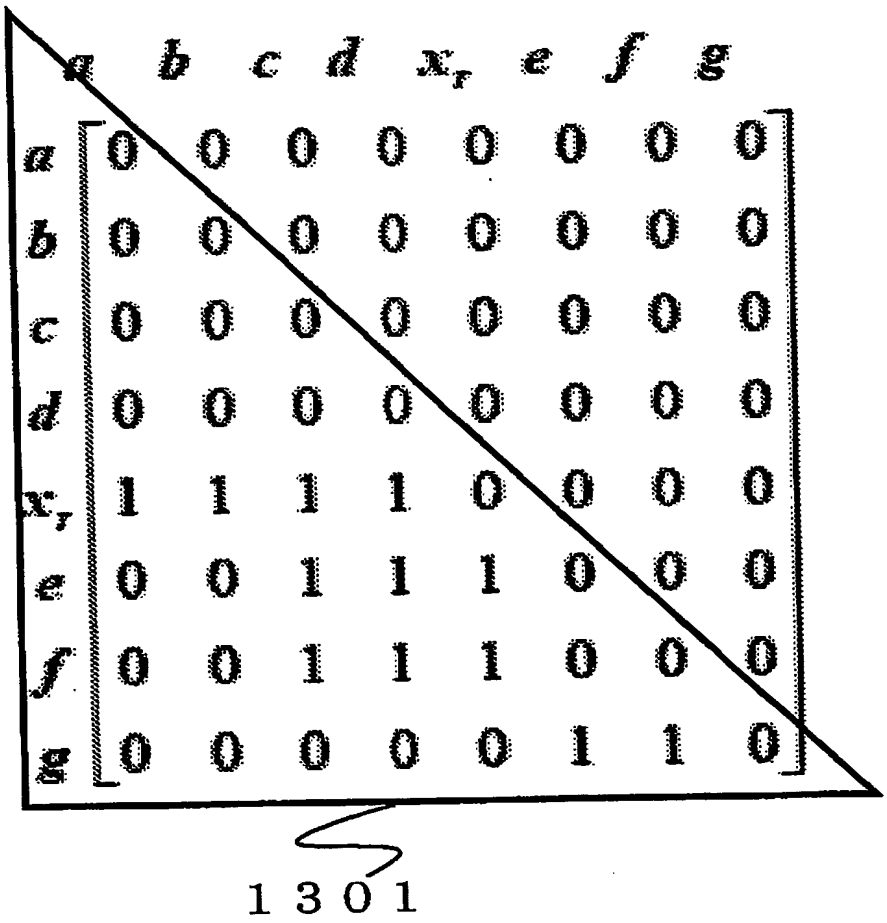
【図１１】


904
902
901
e＝真のとき
g＝e
f＝真のとき
g＝f
e＝c×d
e＝c＋d
a
b
c
d
e
f
g
x_r
BS2
BS3
BS1（出力）
BS1（入力）
903
a＝真のときXr＝BS2
b＝真のときXr＝BS3

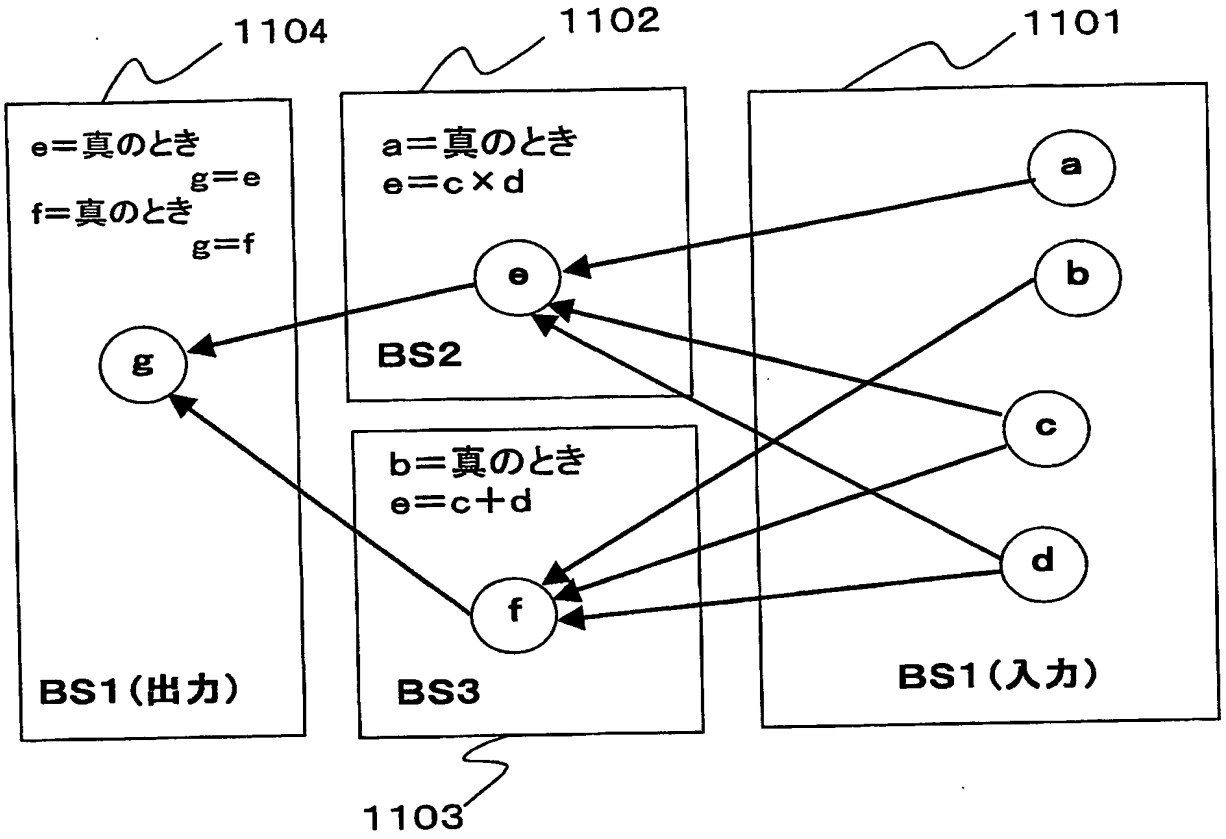
【図１２】


1001
1006
a b c d x_r e f g
a 0 0 0 0 0 0 0 0
b 0 0 0 0 0 0 0 0
c 0 0 0 0 0 0 0 0
d 0 0 0 0 0 0 0 0
x_r 1 1 1 1 0 0 0 0
e 0 0 1 1 1 0 0 0
f 0 0 1 1 1 0 0 0
g 0 0 0 0 0 1 1 0
1007
1002
1004
1008
1005
1003

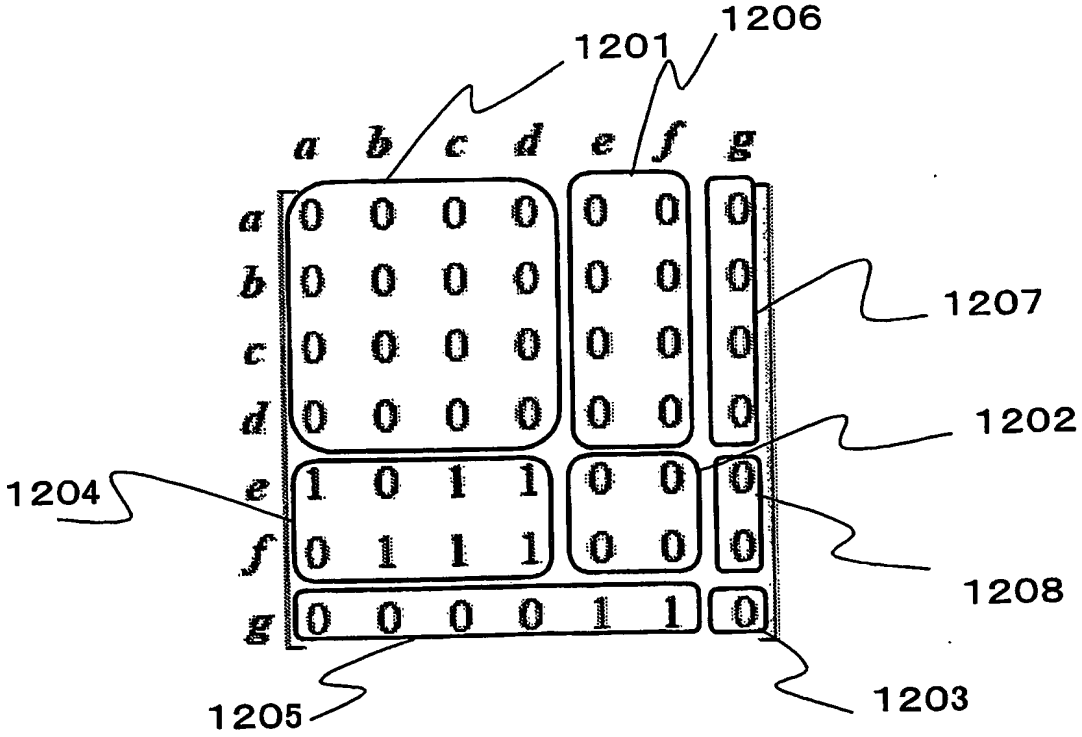
【図１３】


a b c d x_r e f g
a 0 0 0 0 0 0 0 0
b 0 0 0 0 0 0 0 0
c 0 0 0 0 0 0 0 0
d 0 0 0 0 0 0 0 0
x_r 1 1 1 1 0 0 0 0
e 0 0 1 1 1 0 0 0
f 0 0 1 1 1 0 0 0
g 0 0 0 0 0 1 1 0
1301

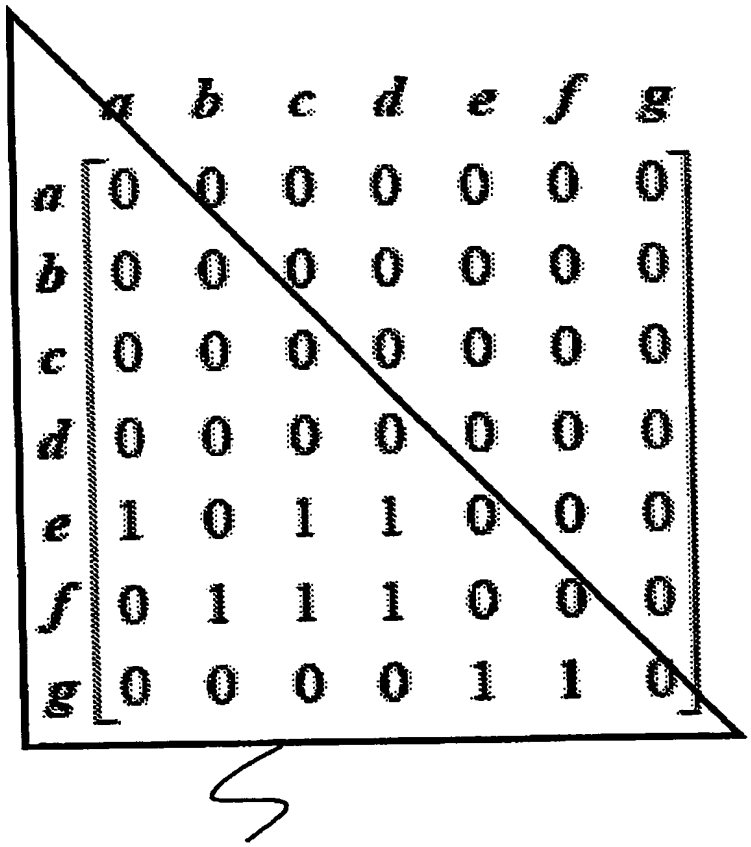
【図１４】


1104
1102
1101
e＝真のとき
g＝e
f＝真のとき
g＝f
a＝真のとき
e＝c×d
b＝真のとき
e＝c＋d
a
b
c
d
e
f
g
BS2
BS1（出力）
BS3
BS1（入力）
1103

【図１５】


1201
1206
a b c d e f g
a 0 0 0 0 0 0 0
b 0 0 0 0 0 0 0
c 0 0 0 0 0 0 0
d 0 0 0 0 0 0 0
e 1 0 1 1 0 0 0
f 0 1 1 1 0 0 0
g 0 0 0 0 1 1 0
1207
1202
1204
1208
1205
1203

【図１６】


a b c d e f g
a 0 0 0 0 0 0 0
b 0 0 0 0 0 0 0
c 0 0 0 0 0 0 0
d 0 0 0 0 0 0 0
e 1 0 1 1 0 0 0
f 0 1 1 1 0 0 0
g 0 0 0 0 1 1 0
1601

【書類名】要約書
【要約】
【課題】
単語単位のプログラムからなるシステムにおける、出力データを生成するための単語単位プログラムの処理を、無駄な繰り返しを排除して最少実行回数で完了すること。
【解決手段】
経路作用要素も単語の１つみなし、単語の関係を定義している単語単位のプログラムとして、要件を定義する。この経路作用要素の定義式実行条件を、その条件が成立するときに経路作用要素が指定する基本構造に属する単語の定義式実行条件に置くことによって経路作用要素を排除し、システム全体の構造を１つにまとめる。こうして得られた単一化されたシステム全体の単語単位のプログラム群（経路作用要素を含めない）に対して、トポロジカルソートを行って単語単位のプログラム群を最適順序に並び替える。これにより、例えば無駄な繰返しを回避することができる。
【選択図】　図１４

特願２００３－３４６４４２

出　願　人　履　歴　情　報

識別番号　［３９６０２３３６２］

１．変更年月日　１９９６年１０月１６日
［変更理由］　新規登録
住　所　東京都江東区潮見二丁目１０番２４号
氏　名　カテナ株式会社